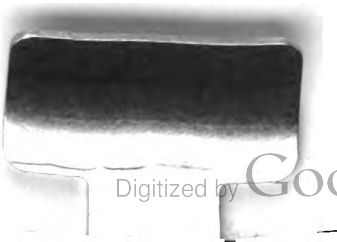


KR2000

Principles of Knowledge Representation and Reasoning

Proceedings of the Seventh International Conference

Breckenridge, Colorado, USA
April 11–15, 2000

Edited by:

Anthony G. Cohn

Fausto Giunchiglia

Bart Selman

Principles of Knowledge Representation and Reasoning:

Proceedings of the Seventh International Conference (KR2000)

The Morgan Kaufmann Series in Representation and Reasoning

Series editor, Ronald J. Brachman (AT&T Bell Laboratories)

Books

James Allen, James Hendler, and Austin Tate, editors
Readings in Planning (1990) ISBN 1-55860-130-9

James F. Allen, Henry A. Kautz, Richard N. Pelavin, and Josh D. Tenenberg
Reasoning About Plans (1991) ISBN 1-55860-137-6

Ronald J. Brachman and Hector Levesque, editors
Readings in Knowledge Representation (1985) ISBN 0-934613-01-X

John Sowa, editor
Principles of Semantic Networks: Explorations in the Representation of Knowledge (1991) ISBN 1-55860-088-4

Ernest Davis
Representations of Commonsense Knowledge (1990) ISBN 1-55860-033-7

Thomas L. Dean and Michael P. Wellman
Planning and Control (1991) ISBN 1-55860-209-7

Janet Kolodner
Case-Based Reasoning (1993) ISBN 1-55860-237-2

Judea Pearl
Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference (1988) ISBN 1-55860-479-0

Daniel S. Weld and Johan de Kleer, editors
Readings in Qualitative Reasoning about Physical Systems (1990) ISBN 1-55860-095-7

David E. Wilkins
Practical Planning: Extending the Classical AI Paradigm (1988) ISBN 0-934613-94-X

Proceedings & Journals

Principles of Knowledge Representation & Reasoning: Proceedings of the First International Conference (KR '89)
Edited by Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter ISBN 1-55860-032-9

Principles of Knowledge Representation & Reasoning: Proceedings of the Second International Conference (KR '91)
Edited by James Allen, Richard Fikes, and Erik Sandewall ISBN 1-55860-165-1

Principles of Knowledge Representation & Reasoning: Proceedings of the Third International Conference (KR '92)
Edited by Bernhard Nebel, Charles Rich, and William Swartout ISBN 1-55860-262-3

Principles of Knowledge Representation & Reasoning: Proceedings of the Fifth International Conference (KR '96)
Edited by Luigia Carlucci Aiello, Jon Doyle, and Stuart C. Shapiro ISBN 1-55860-421-9

Principles of Knowledge Representation & Reasoning: Proceedings of the Sixth International Conference (KR '98)
Edited by Anthony G. Cohn, Lenhart Schubert, and Stuart C. Shapiro ISBN 1-55860-554-1

Reasoning About Actions and Plans: Proceedings of the 1986 Workshop (1987)
Edited by Michael P. Georgeff and Amy L. Lansky ISBN 0-934613-30-3

Theoretical Aspects of Reasoning and Knowledge: Proceedings of the Second Conference (TARK 1988)
Edited by Moshe Y. Vardi ISBN 0-934613-66-4

Theoretical Aspects of Reasoning and Knowledge: Proceedings of the Third Conference (TARK 1990)
Edited by Rohit Parikh ISBN 1-55860-105-8

Theoretical Aspects of Reasoning and Knowledge: Proceedings of the Fourth Conference (TARK 1992)
Edited by Yoram Moses ISBN 1-55860-243-7

Theoretical Aspects of Rationality and Knowledge: Proceedings of the Sixth Conference (TARK 1996)
Edited by Yoav Shoam ISBN 1-55860-417-0

Journal of Artificial Intelligence Research (JAIR): Volumes 2–7, August 1995–April 1998
Edited by Steve Minton and Michael P. Wellman

Principles of Knowledge Representation and Reasoning:

Proceedings of the
Seventh International
Conference on Principles of Knowledge
(KR2000) Representation and Reasoning (7th ...)

Edited by

Anthony G. Cohn
(University of Leeds, UK)

Fausto Giunchiglia
(University of Trento and ITC-irst, Italy)

Bart Selman
(Cornell University, USA)

Breckenridge, Colorado USA
April 12–15, 2000

Sponsored by KR, Inc.

Morgan Kaufmann Publishers, Inc.
San Francisco, California

Sponsoring Editor *Denise E. M. Penrose*
Assistant Developmental Editor *Marilyn Uffer Alan*

These proceedings were managed and produced for the organizers
of the KR2000 conference by Professional Book Center, Denver, Colorado.

The individual papers were submitted in camera-ready form by the contributing authors.

Morgan Kaufmann Publishers
340 Pine Street, Sixth Floor
San Francisco, CA 94104

ISBN 1-55860-690-4
ISSN 1046-9567

00 01 02 03 4 3 2 1

Contents

Preface xi

Acknowledgments xii

SPATIAL REASONING

Spatio-temporal representation and reasoning based on RCC-8 (Best Paper Award) 3
Frank Wolter and Michael Zakharyaschev

Spatial Locations via Morpho-Mereology 15
M. Cristani, A. G. Cohn, and B. Bennett

Continuous Motion in Discrete Space 26
Antony Galton

The Representation of Discrete Multi-resolution Spatial Knowledge 38
John G. Stell

NONMONOTONIC REASONING I

Finding Admissible and Preferred Arguments Can be Very Hard 53
Yannis Dimopoulos, Bernhard Nebel, and Francesca Toni

Complexity Results for Default Reasoning from Conditional Knowledge Bases 62
Thomas Eiter and Thomas Lukasiewicz

Uniform semantic treatment of default and autoepistemic logics 74
Marc Denecker, Victor W. Marek, and Miroslaw Truszczyński

Missionaries and Cannibals in the Causal Calculator 85
Vladimir Lifschitz

REPRESENTATION OF ACTION

Narratives as Programs 99
Ray Reiter

Representing the Knowledge of a Robot 109
Michael Thielscher

A Logic Programming Approach to Conflict Resolution in Policy Management 121
Jan Chomicki, Jorge Lobo, and Shamim Naqvi

INTEGRATION OF KNOWLEDGE SOURCES

On the Difference between Merging Knowledge Bases and Combining them 135
Sébastien Konieczny

BReLS: A System for the Integration of Knowledge Bases 145
Paolo Liberatore and Marco Schaerf

Representing and Aggregating Conflicting Beliefs 153
Pedrito Maynard-Reid II and Daniel Lehmann

AUTOMATED REASONING I

On Strongest Necessary and Weakest Sufficient Conditions (Best Paper Award) 167
Fangzhen Lin

Containment of Conjunctive Regular Path Queries with Inverse 176
Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi

Reduction Rules and Universal Variables for First Order Tableaux and DPLL 186
Fabio Massacci

Deciding K using ℵ 198
Andrei Voronkov

UNCERTAINTY

A Compositional Structured Query Approach to Automated Inference 213
Yousri El Fattah and Mark Alan Peot

An Alternative Combination of Bayesian Networks and Description Logics 225
Phillip M. Yelland

Independence in Qualitative Uncertainty Frameworks 235
N. Ben Amor, S. Benferhat, D. Dubois, H. Geffner, and H. Prade

Spatial representation of spatial relationship knowledge 247
Isabelle Bloch

DESCRIPTION LOGICS

Matching in Description Logics with Existential Restrictions 261
Franz Baader and Ralf Küsters

Expressive ABox Reasoning with Number Restrictions, Role Hierarchies, and Transitively Closed Roles 273
Volker Haarslev and Ralf Möller

Reasoning with Axioms: Theory and Practice 285
Ian Horrocks and Stephan Tobies

Rewriting Concepts Using Terminologies 297
Franz Baader, Ralf Küsters, and Ralf Molitor

DIAGNOSIS

Formulating diagnostic problem solving using an action language with narratives and sensing 311
Chitta Baral, Sheila McIlraith, and Tran Cao Son

Anytime Diagnostic Reasoning using Approximate Boolean Constraint Propagation 323
Alan Verberne, Frank van Harmelen, and Annette ten Teije

Generation of Diagnostic Knowledge by Discrete-Event Model Compilation 333
Gianfranco Lamperti and Marina Zanella

An Algorithm for Belief Revision 345
Renata Wassermann

TEMPORAL REASONING I

Two Problems with Reasoning and Acting in Time 355
Haythem O. Ismail and Stuart C. Shapiro

Cyclical and Granular Time Theories as Subsets of the Herbrand Universe 366
Edjard Mota

A Model for Reasoning about Topologic Relations between cyclic intervals 378
Philippe Balbiani and Aomar Osmani

AUTOMATED REASONING II

Partition-Based Logical Reasoning 389
Eyal Amir and Sheila McIlraith

Significant Inferences: Preliminary Report 401
Philippe Besnard and Torsten Schaub

Unfolding Partiality and Disjunctions in Stable Model Semantics 411
Tomi Janhunen, Ilkka Niemelä, Patrik Simons, and Jia-Huai You

LEARNING AND DECISION MAKING

Relational Representations that Facilitate Learning 425
Chad Cumby and Dan Roth

Experimental Results on Learning Soft Constraints 435
Alessandro Biso, Francesca Rossi, and Alessandro Sperduti

Propositional Logic and One-Stage Decision Making 445
Hélène Fargier, Jérôme Lang, and Pierre Marquis

Logical representation of preferences for group decision making 457
Céline Lafage and Jérôme Lang

APPLICATIONS AND SYSTEMS

OntoMorph: A Translation System for Symbolic Knowledge 471
Hans Chalupsky

An Environment for Merging and Testing Large Ontologies 483
Deborah L. McGuinness, Richard Fikes, James Rice, and Steve Wilder

Long-Term Maintainability of Deployed Knowledge Representation Systems 494
Nestor Rychtyckyj and Robert G. Reynolds

Revision: an application in the framework of GIS 505
Eric Würbel, Robert Jeansoulin, and Odile Papini

REPRESENTATION FORMALISMS

Approximate Objects and Approximate Theories 519
John McCarthy

Iterated Belief Change in the Situation Calculus 527
Steven Shapiro, Maurice Pagnucco, Yves Lespérance, and Hector J. Levesque

Ontology-Based Semantics 539
Mihai Ciocoiu and Dana S. Nau

Supporting Automated Deduction in First-Order Modal Logics 547
Angelo Montanari, Alberto Policriti, and Matteo Slanina

TEMPORAL REASONING II

Controllability characterization and checking in Contingent Temporal Constraint Networks 559
Thierry Vidal

The Augmented Interval and Rectangle Networks 571
Jean-François Condotta

On the complexity of reasoning about repeating events 580
Robert A. Morris and Paul Morris

KNOWLEDGE ENGINEERING

Knowledge Patterns 591
Peter Clark, John Thompson, and Bruce Porter

Knowledge Engineering by Large-Scale Knowledge Reuse—Experience from the Medical Domain 601
Stefan Schulz and Udo Hahn

A Logic Based Language for Parametric Inheritance 611
Hasan M. Jamil

NONMONOTONIC REASONING II

In search of the right extension 625
Jérôme Lang and Pierre Marquis

Ordering explanations and the structural rules for abduction 637
Ramón Pino-Pérez and Carlos Uzcátegui

Valuation-ranked Preferential Model 647
Zhaohui Zhu, Bin Li, Shifu Cheng, and Wujia Zhu

PLANNING (joint program with AIPS 2000)

Planning as Satisfiability with Expressive Action Languages: Concurrency, Constraints and Nondeterminism 657
Enrico Giunchiglia

Learning generalized policies in planning using concept languages 667
Mario Martín and Héctor Geffner

Planning with sensing, concurrency, and exogenous events: logical framework and implementation 678
Luca Iocchi, Daniele Nardi, and Riccardo Rosati

Satisfiability Algorithms and Finite Quantification 690
Matthew L. Ginsberg and Andrew J. Parkes

Desires and Defaults: A Framework for Planning with Inferred Goals 702
Richmond H. Thomason

PANEL ABSTRACTS

Practical Knowledge Representation and the DARPA High Performance Knowledge Bases Project 717
Adam Pease, Vinay Chaudhri, Fritz Lehmann, and Adam Farquhar

Teaching Knowledge Representation: Challenges and Proposals 725
Leora Morgenstern and Richmond H. Thomason

Author Index 735

Preface

This volume contains the papers presented at the Seventh International Conference on Principles of Knowledge Representation and Reasoning. The KR conferences have established themselves as the leading forum for timely, in-depth presentation of progress in the theory and principles underlying the representation and computational manipulation of knowledge.

This is the first KR conference of the new millennium. The traditional very high standard of papers has been maintained and possibly improved. We received 172 extended abstracts and accepted 62. Also included in the proceedings are two papers describing the contents of the two panels being held during KR2000.

A best paper award (sponsored by Morgan Kaufmann Publishers) has been made jointly to two papers: "Spatio-temporal Representation and Reasoning Based on RCC-8" by Frank Wolter and Michael Zakharyaschev, and "On Strongest Necessary and Weakest Sufficient Conditions" by Fangzhen Lin.

The field of KR&R, at least as presented at this conference, seems more diverse than in previous years, with papers in many different areas, with some workshops in the mainstream KR areas (The Eighth International Workshop on Nonmonotonic Reasoning (NMR2000), Practical Reasoning and Rationality, Semantic Approximation, Granularity, and Vagueness) but also one workshop on the emerging applications (The Fourth CollECTeR Conference on Electronic Commerce).

AI has seen increasing fragmentation over recent years as new subfields and their related conferences emerge. However many links among these subfields remain and the conference committee for KR2000 was keen to provide an opportunity for synergies among these fields to emerge. As a result, we negotiated to colocate with the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2000) and to share a day of the technical program: all of the papers, which in the proceedings are collected under the heading "Planning," were presented in the morning, and KR delegates were able to hear KR-related papers from AIPS program in the afternoon. Drew McDermott was jointly invited to speak by KR2000 and AIPS 2000 and opened the shared day.

We are delighted that we also had three other invited speakers. Just as the final day was devoted to making links with planning, each of these invited speakers illuminated links between KR and three other related fields: databases, uncertainty, and robotics. The first was Philip A. Bernstein with a talk entitled "Using Meta-Data to Conquer Database Complexity"; the second was Daphne Koller who spoke on "Probabilistic relational models: Probability meets KR"; and the third was Malik Ghallab with a talk entitled "Planning in Robotics: an Integration Perspective."

Tony Cohn, Conference Chair

Fausto Giunchiglia, Program Co-chair

Bart Selman, Program Co-chair

Acknowledgments

KR2000 would not have been possible without the efforts of a great number of dedicated people. We thank all the program committee members who provided thoughtful and detailed reviews, not only in the first stage of the reviewing process, but also in the second stage when reviews were exchanged and discussed and in the third stage when third or sometimes even fourth reviews were sought, often at very short notice. Our sincere thanks to them and to the aditional reviewers listed below who helped. Fausto Giunchiglia would also like to thank his secretary Carola Dori for her invaluable assistance. Without her it would have been almost impossible.

Program Committee Members

Luigia Carlucci Aiello
U Rome "La Sapienza," Italy

Franz Baader
RWTH Aachen, Germany

Fahiem Bacchus
U Toronto, Canada

Susanne Biundo
U Ulm, Germany

Alexander Bochman
Holon Inst. of Tech., Israel

Alex Borgida
Rutgers U, USA

Gerhard Brewka
U Leipzig, Germany

Marco Cadoli
U Rome "La Sapienza," Italy

Rina Dechter
ICS–U California, Irvine, USA

Didier Dubois
IRIT–CNRS, France

Luis Farinas del Cerro
IRIT–CNRS, France

Richard E. Fikes
Stanford U, USA

Maria Fox
U Durham, UK

Hector Geffner
U Simon Bolivar, VE

Yolanda Gil
ISI/USC, USA

Matthew L. Ginsberg
CIRL, U Oregon, USA

Georg Gottlob
Technische U Wien, Austria

Pat Hayes
U West Florida, USA

Andreas Herzig
IRIT–CNRS, France

Ian Horrocks
U Manchester, UK

Eric Horvitz
Microsoft Research, USA

Henry Kautz
AT&T Labs, USA

Gerhard Lakemeyer
RWTH Aachen, Germany

Maurizio Lenzerini
U Rome "La Sapienza," Italy

Vladimir Lifschitz
U Tex., Austin, USA

Fangzhen Lin
Hong Kong U of Sci. and Tech.

John McCarthy
Stanford U, USA

Nicola Muscettola
NASA Ames Res. Ctr, USA

Karen L. Myers
SRI International, USA

John Mylopolous
U Toronto, Canada

Wolfgang Nejdl
U Hannover, Germany

Hans Jürgen Ohlbach
King's College, UK

Jussi Rintanen
U Freiburg, Germany

Dan Roth
U Illinois, Urbana/Champaign, USA

Erik Sandewall
Linköping U, Sweden

Roberto Sebastiani
U Trento, Italy

Murray Shanahan
Imperial College, UK

Carles Sierra
IIIA–CSIC, Spain

Sam Steel
U Essex, UK

Richmond H. Thomason
U Michigan, USA

Paolo Traverso
ITC-IRST, Italy

Toby Walsh
U York, UK

Additional Reviewers

Jaume Agusti
Matthias Baaz
Mathias Bauer
Robert Baumgartner
Massimo Benerecetti
Brandon Bennett
Guido Boella
Piero Bonatti
Maria Bonet
Richard Booth
Daniel Borrajo
Paolo Bouquet
Diego Calvanese
Marcos A. Castilho
Ernest Davis
Giuseppe De Giacomo
Robert Demolombe
Juergen Dix
Christoph Dornheim
Gonzalo Escalada-Imaz
Wolfgang Faber
Enrico Franconi
Chiara Ghidini
Enrico Giunchiglia
Carole Goble
Lluis Godo
Henrik Grosskreutz
Ullrich Hustadt
Gero Iwan
Kalev Kask
Ralf Kuesters
Nicola Leone
Paolo Liberatore
Thomas Lukasiewicz
Carsten Lutz
Pascal Matsakis
Rob Miller
Ralf Molitor
Angelo Montanari
Bernhard Nebel
Charlie Ortiz
Viviana Patti
Pino Perez
Enric Plaza
Alberto Policriti
Jochen Renz
Michael M. Richter
Omar Rifi
Riccardo Rosati
Tom Russ
Ulrike Sattler
Eddie Schwalb
Luciano Serafini
Wolfgang Slany
Luca Spalazzi
Sergio Tessaris
Stephan Tobies
Hans Tompitz
Hudson Turner
Laure Vieu
Andrei Voronkov
Franz Wotawa

We are also indebted to all people involved in the organization: Deborah L. McGuinness (Local Arrangements Chair), Mary-Anne Williams (Workshops Coordination Chair), Peter Patel-Schneider (Publicity Chair), Neal Lesh (Treasurer).

We also thank those who have financially supported the conference: Bell Labs, IJCAII, MERL, and Morgan Kaufmann Publishers. We also thank the people of Professional Book Center for their help in producing these proceedings.

Finally, we thank the four invited speakers for accepting our invitations, and the efforts they have expended to set the tone of the conference.

Spatial Reasoning

Spatio-temporal representation and reasoning based on RCC-8

Frank Wolter
Institut für Informatik
Universität Leipzig
Augustus-Platz 10-11
04109 Leipzig, Germany
e-mail: wolter@informatik.uni-leipzig.de

Michael Zakharyaschev
Division of Artificial Intelligence
School of Computer Studies
University of Leeds
Leeds LS2 9JT, England
email: mz@scs.leeds.ac.uk

Abstract

We introduce a hierarchy of languages intended for qualitative spatio-temporal representation and reasoning, provide these languages with a topological temporal semantics, construct effective reasoning algorithms, and bound their complexity. The languages are combinations of the well-known fragment RCC-8 of region connection calculus RCC and the point-based temporal logic with the operators 'Since' and 'Until' interpreted on the flow of time $\langle \mathbb{N}, < \rangle$.

1 INTRODUCTION

Qualitative representation and reasoning—as a field within AI—has been quite successful in dealing with both time and space. There exists a wide spectrum of temporal languages (see e.g. (Allen 1983, Gabbay *et al.* 1994, van Benthem 1996)). There is also a variety of spatial formalisms (e.g. (Clarke 1981, Randell *et al.* 1992, Clementini *et al.* 1994, Cohn 1997, Lemon and Pratt 1998)). In both cases effective reasoning procedures have been developed and implemented (e.g. (Plaisted 1986, Kesten *et al.* 1993, Bennett 1994, Fisher 1994, Haarslev *et al.* 1999, Renz and Nebel 1999)). The next apparent and natural step would be to combine these two kinds of reasoning.

The importance of such a step for both theory and applications is beyond any doubt. And of course there have been attempts to construct spatio-temporal hybrids. For example, Clarke's (1981, 1985) intended interpretation of his region-based calculus was spatio-temporal. Region Connection Calculus RCC of (Randell *et al.* 1992) contained a function $\mathsf{space}(x, t)$ for representing the space occupied by object x at moment of time t. Muller (1998) developed a first-order theory for reasoning about motion of spatial entities. However, in contrast to the 'one-dimensional' temporal and spatial cases, no *effective* procedures capable of reasoning about space in time have been developed.

The main aim of this paper is to introduce a hierarchy of languages intended for qualitative spatio-temporal representation and reasoning, provide these languages with topological temporal semantics, construct effective reasoning algorithms, and bound their computational complexity.

The languages we propose are combinations of two well-known and well-understood formalisms in temporal and spatial reasoning. The spatial component is the fragment RCC-8 of RCC containing eight jointly exhaustive and pairwise disjoint base relations between spatial regions. This fragment has attracted considerable attention of the spatial reasoning community (Bennett 1994, 1995, Jonsson and Drakengren 1997, Renz 1998, Renz and Nebel 1998, 1999). First, it may be sufficiently expressive for various application purposes, say in GIS (Egenhofer and Mark 1995, Bennett *et al.* 1997, Haarslev *et al.* 1999). And second, the constraint language of RCC-8 has nice computational properties: it turns out to be decidable (Bennett 1994), in fact NP-complete (Renz and Nebel 1999). Actually, the latter results were obtained by means of encoding RCC-8 in the well-known propositional modal logic S4 (whose necessity operator can be interpreted as the interior operator in topological spaces) extended with the universal modality. This makes natural the choice of the temporal component—the point-based propositional temporal logic PTL with the operators 'Since' and 'Until' interpreted on the flow of time $\langle \mathbb{N}, < \rangle$, rather than Allen's interval calculus which is spiritually closer to RCC. (We hope the examples below and the obtained results will convince the reader that these two formalisms fit together perfectly well indeed.)

PTL is one of the best known temporal logics which has found many applications in CS and AI (e.g. program verification and specification (Manna and Pnueli 1992, 1995), distributed and multi-agent systems (Fagin *et al.* 1995), or temporal databases (Chomicki 1994)); it is decidable and PSPACE-complete (see e.g. (Gabbay *et al.* 1994)). Thus, the problem of constructing effective spatio-temporal formalisms can be viewed as designing decidable two-dimensional modal logics one dimension of which is a topological space and another one the flow of time $\langle \mathbb{N}, < \rangle$.[1] The idea of using such multi-dimensional modal logics for spatio-temporal reasoning has recently been advocated by Bennett and Cohn (1999).

The computational behaviour of a combined spatio-temporal logic depends (i) on the choice of spatial and temporal operators, and (ii) on the degree of the permitted interaction between them. These two parameters give rise to a hierarchy of possible languages.

The simplest one, $\mathcal{ST}_0$, allows applications of the temporal operators Since and Until (as well as the Booleans) only to RCC-8 formulas. (Actually even this language is enough to express, for instance, the condition that change is continuous, or the notion of conceptual neighbourhoods; see e.g. (Cohn 1997).) The most expressive one, $\mathcal{ST}_2^+$, makes it possible to form unions, intersections, and complementations of spatial regions, and to apply temporal operators to both formulas and region terms (for instance, $\bigcirc X$ denotes the state of region X 'tomorrow').

The natural semantics for these languages are temporal models for PTL in which each state is some fixed topological space. We prove that the satisfiability problem for all our languages in topological temporal models is decidable, with the computational complexity ranging from NP to EXPSPACE. We also study the structure of the simplest topological spaces that are enough to satisfy all satisfiable $\mathcal{ST}_2^+$-formulas, and consider the problem of realising spatio-temporal formulas in models based on Euclidean space.

2 REGION CONNECTION CALCULUS

Full RCC. RCC—*Region Connection Calculus*—is a first-order theory designed by Randell, Cui, and Cohn (1992) for qualitative spatial representation and reasoning. The basic language of RCC contains only one primitive predicate $\mathsf{C}(X, Y)$ which is read 'region X *is connected* with region Y'. Starting from this, one can define other kinds of relations between spatial regions. The basic ones are: $\mathsf{DC}(X, Y)$ (X and Y are disconnected), $\mathsf{EC}(X, Y)$ (X is externally connected to Y), $\mathsf{PO}(X, Y)$ (X partially overlaps Y), $\mathsf{EQ}(X, Y)$ (X is identical with Y), $\mathsf{TPP}(X, Y)$ (X is a tangential proper part of Y), $\mathsf{NTPP}(X, Y)$ (X is a nontangential proper part of Y), and the inverses $\mathsf{TPPi}(X, Y)$ and $\mathsf{NTPPi}(X, Y)$ of the last two.

The intended models of RCC are topological spaces $\mathfrak{T} = \langle U, \mathbb{I} \rangle$, where U is the *universe* of the space and $\mathbb{I}$ an *interior operator* on U satisfying the usual Kuratowski axioms. Individual variables of RCC range over non-empty regular closed sets of $\mathfrak{T}$, i.e., an *assignment* in $\mathfrak{T}$ is a map $\mathfrak{a}$ associating with every variable X a set $\mathfrak{a}(X) \subseteq U$ such that $\mathfrak{a}(X) \neq \emptyset$ and $\mathfrak{a}(X) = \mathbb{CI}\,\mathfrak{a}(X)$, where $\mathbb{C}$ is the *closure operator* on U dual to $\mathbb{I}$. The intended meaning of $\mathsf{C}(X, Y)$ is formalised then as follows: $\mathfrak{T} \models^{\mathfrak{a}} \mathsf{C}(X, Y)$ iff $\mathfrak{a}(X) \cap \mathfrak{a}(Y) \neq \emptyset$. From the computational point of view RCC is too expressive: as was observed by Gotts (1996), the full first-order theory of RCC is undecidable. Fortunately, there are various decidable (and even tractable) fragments of RCC. One of the most important is known as RCC-8.

RCC-8. If we are interested only in relationships between spatial regions without taking into account their topological shape, then the eight predicates above are enough: they are jointly exhaustive and pairwise disjoint, which means that any two regions stand precisely in one of these eight relations. Moreover, according to the experiments reported by Renz and Nebel (1998), the eight predicates turn out to be conceptually cognitive adequate in the sense that people indeed distinguish between these relations.

Formally, the language of RCC-8 consists of a set of individual variables, called *region variables*, the eight binary predicates DC, EC, PO, EQ, TPP, TPPi, NTPP, NTPPi, and the Booleans out of which we can construct in the usual way *spatial formulas*. The main reasoning task for RCC-8 can be formulated as follows:

Given a finite set Σ of spatial formulas, decide whether Σ is satisfiable in a topological space, i.e., whether there exist a topological space $\mathfrak{T}$ and an assignment $\mathfrak{a}$ in it such that $\mathfrak{T} \models^{\mathfrak{a}} \Sigma$.

That this *satisfiability problem* is decidable was shown by Bennett (1994, 1996) who encoded RCC-8 in propositional intuitionistic logic and modal system S4 using the well-known fact that both of them are complete with respect to topological spaces (Stone 1937, Tarski 1938); an elementary proof of the correctness of this encoding is provided in (Nutt 1999). Renz and Nebel (1999) and Renz (1999) proved that the satisfiability

[1] Actually, the obtained results can be generalised to some other flows of time, for instance $\langle \mathbf{Z}, < \rangle$ or $\langle \mathbf{R}, < \rangle$.

problem for RCC-8 formulas is actually NP-complete and described all maximal tractable fragments of RCC-8.

3 TEMPORALISING RCC-8

Suppose now that the spatial configuration we are interested in is changing over time. Let us imagine, for instance, that the flow of time is $\langle \mathbb{N}, < \rangle$ and that we have the temporal operators $\mathcal{S}$ (Since) and $\mathcal{U}$ (Until); other standard operators like $\bigcirc$ (at the next moment), $\Box^+$ (always in the future), $\Diamond^+$ (some time in the future), etc. are defined via $\mathcal{S}$ and $\mathcal{U}$ in the usual way. We will assume also that space itself with its topology always remains the same. However, the spatial regions occupied by the objects under consideration may change with time passing by. This naïve picture is formalised by the following concept of topological temporal model.

Definition 1 A *topological temporal model* (or *tt-model*) based on a topological space $\mathfrak{T} = \langle U, \mathbb{I} \rangle$ is a triple of the form $\mathfrak{M} = \langle \mathfrak{T}, \mathbb{N}, \mathfrak{a} \rangle$, where $\mathfrak{a}$, an *assignment* in $\mathfrak{T}$, associates with every region variable X and every moment of time $n \in \mathbb{N}$ a set $\mathfrak{a}(X, n) \subseteq U$ such that $\mathfrak{a}(X, n) = \mathbb{C}\mathbb{I}\mathfrak{a}(X, n)$ and $\mathfrak{a}(X, n) \neq \emptyset$. For each n, we take $\mathfrak{a}_n$ to be the function defined by $\mathfrak{a}_n(X) = \mathfrak{a}(X, n)$.

There are different ways of introducing a temporal dimension into the syntax of RCC-8.

$\mathcal{ST}_0$. The most obvious one is to allow applications of the operators $\mathcal{S}$ and $\mathcal{U}$ (along with the Booleans) to spatial formulas. Denote the resulting *spatio-temporal language* by $\mathcal{ST}_0$.

Definition 2 For a tt-model $\mathfrak{M} = \langle \mathfrak{T}, \mathbb{N}, \mathfrak{a} \rangle$, an $\mathcal{ST}_0$-formula φ, and $n \in \mathbb{N}$, define the *truth-relation* $(\mathfrak{M}, n) \models \varphi$ — 'φ holds in $\mathfrak{M}$ at moment n' — by induction on the construction of φ:

- if φ contains no temporal operators, then $(\mathfrak{M}, n) \models \varphi$ iff $\mathfrak{T} \models^{\mathfrak{a}_n} \varphi$;
- $(\mathfrak{M}, n) \models \varphi \mathcal{U} \psi$ iff there is $k > n$ such that $(\mathfrak{M}, k) \models \psi$ and $(\mathfrak{M}, l) \models \varphi$ for every $l \in (n, k)$, where $(n, k) = \{m : n < m < k\}$;
- $(\mathfrak{M}, n) \models \varphi \mathcal{S} \psi$ iff there is $k < n$ such that $(\mathfrak{M}, k) \models \psi$ and $(\mathfrak{M}, l) \models \varphi$ for every $l \in (k, n)$.

The interaction between time and space in $\mathcal{ST}_0$ is rather weak, and it is not hard to show that the following theorem holds. (For proofs of this and further theorems see Sections 4 and 5 and the full paper at `http://www.informatik.uni-leipzig.de/~wolter`.)

Theorem 3 *The satisfiability problem for $\mathcal{ST}_0$-formulas in tt-models is PSPACE-complete.*

The language $\mathcal{ST}_0$ is expressive enough to capture *continuity of changes* (see e.g. (Cohn 1997)):

$$\Box^+(\mathsf{DC}(X,Y) \to \bigcirc(\mathsf{DC}(X,Y) \vee \mathsf{EC}(X,Y))),$$
$$\Box^+(\mathsf{EC}(X,Y) \to \bigcirc(\mathsf{DC}(X,Y) \vee \mathsf{EC}(X,Y) \vee \mathsf{PO}(X,Y))),$$
$$\Box^+(\mathsf{PO}(X,Y) \to \bigcirc(\mathsf{EC}(X,Y) \vee \mathsf{PO}(X,Y) \vee \mathsf{TPP}(X,Y) \vee \mathsf{EQ}(X,Y) \vee \mathsf{TPP}^{-1}(X,Y))),$$

etc.

The first of these formulas, for instance, says that if two regions are disconnected at some moment, then at the next moment they either will remain disconnected or will be externally connected (but will not overlap).

However, the expressive power of $\mathcal{ST}_0$ is rather limited. In particular, we can compare regions only at one moment of time, but we are not able to connect a region as it is today with its state tomorrow to say e.g. that it is expanding or remains the same. In other words, we can express the dynamics of relations between regions, say, $\neg\Box^+\mathsf{P}(Kosovo, Yugoslavia)$ ('it is not true that Kosovo will always be part of Yugoslavia') but not the dynamics of regions themselves, e.g. that $\Box^+\mathsf{P}(EU, \bigcirc EU)$, where $\bigcirc EU$ at moment n denotes the space occupied by the EU at moment $n+1$ (so the last formula means: 'the EU will never contract'). This new construct can also be used to refine the continuity assumption by requiring that $\Box^+(\mathsf{EQ}(X, \bigcirc X) \vee \mathsf{O}(X, \bigcirc X))$, i.e., 'regions X and $\bigcirc X$ either coincide or overlap.'

$\mathcal{ST}_1$. To capture this dynamics, we extend $\mathcal{ST}_0$ by allowing applications of the next-time operator $\bigcirc$ not only to formulas but also to region variables. Thus, arguments of the RCC-8 predicates can now be region variables X prefixed by arbitrarily long sequences of $\bigcirc$, say $\bigcirc\bigcirc X$ (representing region X as it will be the day after tomorrow). Denote the resulting language by $\mathcal{ST}_1$, and let $\mathcal{ST}_1'$ be its sublanguage with only one temporal operator $\bigcirc$. Definition 2 is extended by the following clause: $\mathfrak{a}(\bigcirc X, n) = \mathfrak{a}(X, n+1)$.

Theorem 4 (i) *The satisfiability problem for $\mathcal{ST}_1$-formulas in tt-models is decidable in EXPSPACE.*

(ii) *The satisfiability problem for $\mathcal{ST}_1'$-formulas is NP-complete.*

Using $\mathcal{ST}_1$ we can express, say, that region X will always be the same (i.e., X is *rigid*): $\Box^+\mathsf{EQ}(X, \bigcirc X)$, or that it has at most two distinct states, one on even days, another on odd ones: $\Box^+\mathsf{EQ}(X, \bigcirc\bigcirc X)$. Note, by the way, that the $\mathcal{ST}_1$-formula $\Box^+\mathsf{NTPP}(X, \bigcirc X)$ is

satisfiable only in models based on infinite topological spaces—unlike $\mathcal{ST}_0$- (and of course, RCC-8) formulas for which finite topological spaces are always sufficient.

It may appear that $\mathcal{ST}_1$ is able to compare regions only within fixed time intervals. However, using an auxiliary rigid variable X we can write, for instance, $\Box^+\mathsf{EQ}(X, \bigcirc X) \wedge \Diamond^+\mathsf{EQ}(X, EU) \wedge \mathsf{P}(Russia, X)$. This formula is satisfiable iff 'someday in the future the *present* territory of Russia will be part of the EU.' Note that the formula $\Diamond^+\mathsf{P}(Russia, EU)$ means that there will be a day when Russia—its territory on that day (say, without Chechnya but with Byelorussia)—becomes part of the EU.

Imagine now that we want to express in our spatio-temporal language that all countries in Europe will pass through the Euro-zone, but only Germany will use the euro forever. Unfortunately, we don't know which countries will be formed in Europe in the future, so we can't simply write down all formulas of the form $\Diamond^+\mathsf{P}(X, Euro\text{-}zone)$. Hopefully, Europe is rigid, and so what we need is the possibility of constructing regions $\Diamond^+X$ and $\Box^+X$ which contain all the points that will belong to region X in the future, and only common points of all future states of X, respectively. Then we can write: $\mathsf{EQ}(Europe, \Diamond^+ Euro\text{-}zone)$ and $\mathsf{EQ}(Germany, \Box^+ Euro\text{-}zone)$. The formula $\mathsf{P}(Russia, \Diamond^+ EU)$ says that all points of the present territory of Russia will belong to the EU in the future (but perhaps at different moments of time).

$\mathcal{ST}_2$. So let us allow applications of the temporal operators $\bigcirc$, $\Box^+$, $\Diamond^+$ (and possibly their past counterparts) to region variables.

Definition 5 Every region variable is a *region term*. If t is a *region term* then so are $\bigcirc t$, $\Box^+ t$, and $\Diamond^+ t$.

Denote by $\mathcal{ST}_2$ the language that results from $\mathcal{ST}_0$ by allowing the use of region terms. The intended semantics for region terms is defined as follows.

Definition 6 Let $\mathfrak{M} = \langle \mathfrak{T}, \mathbb{N}, \mathfrak{a} \rangle$ be a tt-model. Define by induction the *value* $\mathfrak{a}(t, n)$ of a region term t under $\mathfrak{a}$ at n in $\mathfrak{M}$: $\mathfrak{a}(\bigcirc t, n) = \mathfrak{a}(t, n+1)$, $\mathfrak{a}(\Box^+ t, n) = \mathbb{CI} \bigcap_{k>n} \mathfrak{a}(t, k)$, and $\mathfrak{a}(\Diamond^+ t, n) = \mathbb{CI} \bigcup_{k>n} \mathfrak{a}(t, k)$.

It is to be noted, however, that this definition uses infinite unions and intersections of regions which are rather problematic. As was observed in (Randell *et al.* 1992), infinite unions contradict the axioms of full RCC. The infinite intersection of a sequence of shrinking regions may result in an empty set which by definition is not a region. And finally, as we shall see below, infinite operations bring various semantical complications. To avoid this problem we can try to restrict assignments in models in such a way that infinite intersections and unions can be reduced to finite ones.

There are several ways of doing this. One idea would be to accept the *Finite Change Assumption* (FCA): *No region can change its spatial configuration infinitely often.*

This means that under FCA we consider only those tt-models $\mathfrak{M} = \langle \mathfrak{T}, \mathbb{N}, \mathfrak{a} \rangle$ that satisfy the following condition: for every region term t, there is $n \in \mathbb{N}$ such that $\mathfrak{a}(t, k) = \mathfrak{a}(t, n)$ for all $k > n$. Actually, FCA can be captured by the $\mathcal{ST}_2$-formulas $\Diamond^+\Box^+\mathsf{EQ}(t, \bigcirc t)$. Of course, FCA excludes some mathematically interesting cases. Yet, it is absolutely adequate for many applications, for example, when we are planning a job which eventually must be completed (consider a robot painting a wall). Optimists will accept FCA to describe the geography of Europe in the examples above. In temporal databases the time line is often assumed to be finite, though arbitrarily long, which corresponds to FCA.

Theorem 7 *The satisfiability problem for $\mathcal{ST}_2$-formulas in FCA-models is decidable in EXPSPACE.*

Another, more general, way of reducing infinite unions and intersections to finite ones is to adopt the *Finite State Assumption* (FSA): *Every region can have only finitely many possible states (although it may change its states infinitely often).*

Definition 8 Say that a model $\mathfrak{M} = \langle \mathfrak{T}, \mathbb{N}, \mathfrak{a} \rangle$ is an *FSA-model* if for every region term t there are finitely many regular closed sets $A_1, \ldots, A_m \subseteq U$ such that $\{\mathfrak{a}(t, n) : n \in \mathbb{N}\} = \{A_1, \ldots, A_m\}$.

Theorem 9 (i) *The satisfiability problem for $\mathcal{ST}_2$-formulas in FSA-models is decidable in EXPSPACE.*

(ii) *An $\mathcal{ST}_2$-formula is satisfiable in an FSA-model iff it is satisfiable in an FSA-model based on a finite topological space.*

$\mathcal{ST}_i^+$. We can make our languages $\mathcal{ST}_i$, $i = 0, 1, 2$, even more expressive by allowing applications of the Booleans to region terms. Their semantical meaning is defined as follows: $\mathfrak{a}(t \vee t', n) = \mathfrak{a}(t, n) \cup \mathfrak{a}(t', n)$, $\mathfrak{a}(t \wedge t', n) = \mathbb{CI}(\mathfrak{a}(t, n) \cap \mathfrak{a}(t', n))$, $\mathfrak{a}(\neg t, n) = \mathbb{CI}(U - \mathfrak{a}(t, n))$. So we can write, e.g.

$$\mathsf{EQ}(UK, Great\_Britain \vee Northern\_Ireland).$$

Let $\mathcal{ST}_i^+$ be the resulting family of languages. It turns out that all theorems above remain valid after replacing $\mathcal{ST}_i$ with $\mathcal{ST}_i^+$.

4 MODAL ENCODING OF $\mathcal{ST}_2^+$

The decidability and complexity results formulated above are proved by means of embedding $\mathcal{ST}_2^+$ into a temporalised version of the propositional modal logic S4 extended with the universal modality (Goranko and Passy 1992) and then using the method of quasimodels developed in (Wolter and Zakharyaschev 1999). Denote by $\mathcal{ML}$ the propositional modal language whose connectives are the Booleans, the necessity and possibility operators I and C of S4, the 'universal' necessity and possibility $\forall$ and $\exists$, and the temporal operators $\mathcal{S}$ and $\mathcal{U}$. What is the intended semantics of $\mathcal{ML}$? When encoding pure RCC-8 in S4 with $\forall$, we can use either of two different kinds of models: (more general) topological models of S4 and (more transparent) Kripke models (which is explained, for instance, by the fact that S4 has the finite model property). The addition of the temporal component makes the situation more complicated, for we can interpret $\mathcal{ML}$-formulas in structures of two kinds that are not equivalent with respect to these formulas.

Definition 10 A *Kripke $\mathcal{ML}$-model* is a triple of the form $\mathfrak{K} = \langle \mathfrak{F}, \mathbf{N}, \mathfrak{V} \rangle$ where $\mathfrak{F} = \langle W, R \rangle$ is a quasi-order (a frame for S4) and $\mathfrak{V}$, a *valuation*, is a map associating with every propositional variable p and every $n \in \mathbf{N}$ a subset $\mathfrak{V}(p,n) \subseteq W$. The truth-relation $(u,n) \models_{\mathfrak{K}} \varphi$ in $\mathfrak{K}$ is defined as follows:

- $(u,n) \models_{\mathfrak{K}} p$ iff $u \in \mathfrak{V}(p,n)$, p a propositional variable,
- $(u,n) \models_{\mathfrak{K}} \forall\psi$ iff $(v,n) \models_{\mathfrak{K}} \psi$ for all $v \in W$,
- $(u,n) \models_{\mathfrak{K}} I\psi$ iff $(v,n) \models_{\mathfrak{K}} \psi$ for all $v \in W$ such that uRv,
- $(u,n) \models_{\mathfrak{K}} \psi\mathcal{U}\chi$ iff there is $k > n$ such that $(u,k) \models_{\mathfrak{K}} \chi$ and $(u,m) \models_{\mathfrak{K}} \psi$ for all $m \in (n,k)$,
- $(u,n) \models_{\mathfrak{K}} \psi\mathcal{S}\chi$ iff there is $k < n$ such that $(u,k) \models_{\mathfrak{K}} \chi$ and $(u,m) \models_{\mathfrak{K}} \psi$ for all $m \in (k,n)$,

plus the standard clauses for the Booleans. An $\mathcal{ML}$-formula φ is *satisfied* in $\mathfrak{K}$ if $(u,n) \models_{\mathfrak{K}} \varphi$ for some $u \in W$ and $n \in \mathbf{N}$.

Definition 11 A *topological model* for $\mathcal{ML}$ is the structure $\mathfrak{N} = \langle \mathfrak{T}, \mathbf{N}, \mathfrak{U} \rangle$ in which $\mathfrak{T} = \langle U, \mathbb{I} \rangle$ is a topological space and $\mathfrak{U}$, a *valuation*, is a map associating with every propositional variable p and every $n \in \mathbf{N}$ a set $\mathfrak{U}(p,n) \subseteq U$. $\mathfrak{U}$ is then extended to arbitrary $\mathcal{ML}$-formulas in the following way:

- $\mathfrak{U}(\psi \wedge \chi, n) = \mathfrak{U}(\psi,n) \cap \mathfrak{U}(\chi,n)$,
- $\mathfrak{U}(\neg\psi, n) = U - \mathfrak{U}(\psi,n)$,
- $\mathfrak{U}(\forall\psi, n) = U$ if $\mathfrak{U}(\psi,n) = U$, and $\mathfrak{U}(\forall\psi,n) = \emptyset$ otherwise,
- $\mathfrak{U}(I\psi, n) = \mathbb{I}\mathfrak{U}(\psi,n)$,
- $x \in \mathfrak{U}(\psi\mathcal{U}\chi, n)$ iff there exists $k > n$ such that $x \in \mathfrak{U}(\chi,k)$ and $x \in \mathfrak{U}(\psi,m)$ for all $m \in (n,k)$,
- $x \in \mathfrak{U}(\psi\mathcal{S}\chi, n)$ iff there exists $k < n$ such that $x \in \mathfrak{U}(\chi,k)$ and $x \in \mathfrak{U}(\psi,m)$ for all $m \in (k,n)$.

In particular, we have $\mathfrak{U}(\Diamond^+\psi, n) = \bigcup_{k>n} \mathfrak{U}(\psi,k)$ and $\mathfrak{U}(\Box^+\psi, n) = \bigcap_{k>n} \mathfrak{U}(\psi,k)$. An $\mathcal{ML}$-formula φ is *satisfied* in $\mathfrak{N}$ if $\mathfrak{U}(\varphi,n) \neq \emptyset$ for some $n \in \mathbf{N}$.

The sets of $\mathcal{ML}$-formulas satisfiable in Kripke models and topological models turn out to be different. Of course, every Kripke model (based on $\langle W,R \rangle$) is equivalent to some topological model (with $\langle W, \mathbb{I} \rangle$ as the underlying topological space, where $\mathbb{I}X$ is the maximal R-closed subset of X). But the converse does not hold: the formula $\Diamond^+ Cp \leftrightarrow C\Diamond^+ p$ is valid in every Kripke $\mathcal{ML}$-model, but not in every topological $\mathcal{ML}$-model, because there is a sequence X_n of closed sets in $\mathbb{R}$ such that $\bigcup_{n\in\mathbf{N}} X_n$ is not closed.

Fortunately, the two types of models are equivalent with respect to *modal translations* of $\mathcal{ST}_2^+$-formulas under the finite state assumption FSA. The modal translation $\dagger$ from $\mathcal{ST}_2^+$ into $\mathcal{ML}$ is defined as follows.

Definition 12 For a region term t, define an $\mathcal{ML}$-formula t^* by taking $X_i^* = p_i$, X_i a region variable, $(\bigcirc t)^* = CI \bigcirc t^*$, $(\Diamond^+ t)^* = CI\Diamond^+ t^*$, $(\Box^+ t)^* = CI\Box^+ t^*$, $(t_1 \vee t_2)^* = CI(t_1^* \vee t_2^*)$, $(t_1 \wedge t_2)^* = CI(t_1^* \wedge t_2^*)$, $(\neg t)^* = CI\neg t^*$.

For atomic $\mathcal{ST}_2^+$-formulas, let

$(\mathsf{DC}(t_1,t_2))^* = \neg\exists(t_1^* \wedge t_2^*)$,
$(\mathsf{EC}(t_1,t_2))^* = \exists(t_1^* \wedge t_2^*) \wedge \neg\exists(It_1^* \wedge It_2^*)$,
$(\mathsf{PO}(t_1,t_2))^* = \exists(It_1^* \wedge It_2^*) \wedge \exists(It_1^* \wedge \neg t_2^*) \wedge \exists(\neg t_1^* \wedge It_2^*)$,
$(\mathsf{EQ}(t_1,t_2))^* = \forall(t_1^* \leftrightarrow t_2^*)$,
$(\mathsf{TPP}(t_1,t_2))^* = \forall(\neg t_1^* \vee t_2^*) \wedge \exists(t_1^* \wedge C\neg t_2^*) \wedge \exists(\neg t_1^* \wedge t_2^*)$,
$(\mathsf{TPPi}(t_1,t_2))^* = (\mathsf{TPP}(t_2,t_1))^*$,
$(\mathsf{NTPP}(t_1,t_2))^* = \forall(\neg t_1^* \vee It_2^*) \wedge \exists(\neg t_1^* \wedge t_2^*)$,
$(\mathsf{NTPPi}(t_1,t_2))^* = (\mathsf{NTPP}(t_2,t_1))^*$,
$(\mathsf{NE}(t))^* = \exists t^*$,
$(\mathsf{Regular}(t))^* = \forall(I\neg t^* \vee CIt^*)$.

Suppose now that φ is an arbitrary $\mathcal{ST}_2^+$-formula. Denote by φ^* the result of replacing all occurrences of RCC-8 predicates $R(t_1,t_2)$ in φ by $(R(t_1,t_2))^*$ and then put

$$\varphi^\dagger = \varphi^* \wedge {}^* \bigwedge_{t \in term\varphi} (\mathsf{NE}(t))^* \wedge (\mathsf{Regular}(t))^*,$$

where $term\varphi$ is the set of all region terms occurring in φ. The $\mathcal{ML}$-formula $\varphi^\dagger$ is called the *modal translation* of φ.

Say that a topological $\mathcal{ML}$-model $\langle \mathfrak{T}, \mathbf{N}, \mathfrak{U} \rangle$ (or a Kripke $\mathcal{ML}$-model $\langle \mathfrak{F}, \mathbf{N}, \mathfrak{U} \rangle$) *satisfies FSA* for an $\mathcal{ML}$-formula φ if, for every variable p occurring in φ and every $n \in \mathbf{N}$, there exist finitely many sets $A_1, \dots, A_k$ such that

$$\{\mathfrak{U}(p, m) : m > n\} = \{A_1, \dots, A_k\}.$$

It is easy to show by induction that in this case we also have that for every $\mathcal{ML}$-formula ψ built up from variables in φ and every $n \in \mathbf{N}$ there are sets $B_1, \dots, B_l$ such that

$$\{\mathfrak{U}(\psi, m) : m > n\} = \{B_1, \dots, B_l\}.$$

By a straightforward induction one can prove the following:

Theorem 13 *An $\mathcal{ST}_2^+$-formula is satisfiable in a tt-model (with FSA) iff its modal translation is satisfiable in a topological $\mathcal{ML}$-model (with FSA).*

The modal translations of $\mathcal{ST}_2^+$-formulas form a rather special fragment of the modal language $\mathcal{ML}$. Renz (1998) showed that an RCC-8 formula φ is satisfiable iff $\varphi^\dagger$ is satisfiable in a Kripke model based on an S4-frame of depth ≤ 1 and width ≤ 2 (which means that it contains no chains of more than 2 distinct points, and no point has more than 2 distinct proper successors). It turns out that this result can be generalised to $\mathcal{ST}_2^+$-formulas:

Theorem 14 (i) *An $\mathcal{ST}_2^+$-formula φ is satisfiable in a tt-model with FSA iff $\varphi^\dagger$ is satisfiable in a Kripke $\mathcal{ML}$-model with FSA whose underlying* S4*-frame is of depth ≤ 1 and width ≤ 2.*

(ii) *An $\mathcal{ST}_1^+$-formula φ is satisfiable in a tt-model iff $\varphi^\dagger$ is satisfiable in a Kripke $\mathcal{ML}$-model whose underlying* S4*-frame is of depth ≤ 1 and width ≤ 2.*

(A proof can be found in the full paper.)

5 DECIDABILITY

We begin by considering the logics $\mathcal{ST}_0$ and $\mathcal{ST}_1$. To simplify presentation, we assume that we have only one temporal operator $\mathcal{U}$; the reader should have no problems in extending the proofs to $\mathcal{S}$. Suppose $\varphi \in \mathcal{ST}_1$. Denote by $term(\varphi)$ the set of region terms $\bigcirc^m X$ occurring in φ. And let $sub(\varphi)$ be the set of all subformulas in φ. Without loss of generality we may assume that $sub(\varphi)$ is closed under $\neg$.

Definition 15 (fork) A *fork* is a frame $\mathfrak{f} = \langle W_\mathfrak{f}, R_\mathfrak{f} \rangle$ such that

- $W_\mathfrak{f} = \{a_\mathfrak{f}, b_\mathfrak{f}, c_\mathfrak{f}\}$,
- $a_\mathfrak{f}$ is the root of $\mathfrak{f}$, while $b_\mathfrak{f}$ and $c_\mathfrak{f}$ are its two immediate successors.

Thus $R_\mathfrak{f}$ is the reflexive closure of the relation

$$\{\langle a_\mathfrak{f}, b_\mathfrak{f} \rangle, \langle a_\mathfrak{f}, c_\mathfrak{f} \rangle\}.$$

A *labelled fork* for φ is the pair $\langle \mathfrak{f}, l_\mathfrak{f} \rangle$, where $\mathfrak{f}$ is a fork and $l_\mathfrak{f}$ a *labelling function* which associates with every point x in $\mathfrak{f}$ a subset $l_\mathfrak{f}(x)$ of $term(\varphi)$ in such a way that $t \in l_\mathfrak{f}(x)$ iff there is a final point y in $\mathfrak{f}$ accessible from x with $t \in l_\mathfrak{f}(y)$ (this reflects the fact that all region terms are interpreted by regular closed sets).

A set $\boldsymbol{F}$ of labelled forks can be regarded as a Kripke frame $\langle \boldsymbol{W}, \boldsymbol{R} \rangle$—the disjoint union of forks in $\boldsymbol{F}$—with all points $x \in \boldsymbol{W}$ labelled by $l_{\boldsymbol{F}}(x) \subseteq term(\varphi)$, where $l_{\boldsymbol{F}}(x) = l_\mathfrak{f}(x)$ if $x \in W_\mathfrak{f}$.

For an RCC-8 predicate P and region terms t_1, t_2, we write $\boldsymbol{F} \models P(t_1, t_2)$ iff $P(t_1, t_2)$ holds in the topological space determined by $\boldsymbol{F}$ under the assignment

$$\mathfrak{a}(t) = \{x \in \boldsymbol{W} : t \in l_{\boldsymbol{F}}(x)\}.$$

Definition 16 (type) A *formula type* for φ is a subset Φ of $sub(\varphi)$ such that

- $\psi \wedge \chi \in \Phi$ iff $\psi, \chi \in \Phi$, for every $\psi \wedge \chi \in sub(\varphi)$;
- $\neg\psi \in \Phi$ iff $\psi \notin \Phi$, for every $\psi \in sub(\varphi)$.

Definition 17 (quasistate) We say that the pair $\langle \boldsymbol{F}, \Phi \rangle$ is a *quasistate* for φ if

- $\boldsymbol{F}$ is a set of pairwise non-isomorphic labelled forks, and
- Φ is a formula type for φ such that for every $P(t_1, t_2) \in sub(\varphi)$, we have $P(t_1, t_2) \in \Phi$ iff $\boldsymbol{F} \models P(t_1, t_2)$.

Denote by $\sharp(\varphi)$ the number of pairwise non-isomorphic quasistates for φ. Clearly

$$\sharp(\varphi) \leq 2^{2^{2 \cdot |term(\varphi)|}} \cdot 2^{|sub(\varphi)|}.$$

Definition 18 (suitable pair) Say that an ordered pair $\langle \mathfrak{f}, l_\mathfrak{f} \rangle$, $\langle \mathfrak{g}, l_\mathfrak{g} \rangle$ of labelled forks for φ is *suitable* if there is an isomorphism σ from $\mathfrak{f}$ onto $\mathfrak{g}$ such that, for every $\bigcirc t \in term(\varphi)$ and every $x \in W_\mathfrak{f}$,

$$\bigcirc t \in l_\mathfrak{f}(x) \quad \text{iff} \quad t \in l_\mathfrak{g}(\sigma(x)).$$

An ordered pair $\langle \boldsymbol{F}, \Phi \rangle$, $\langle \boldsymbol{G}, \Psi \rangle$ of quasistates for φ is called *suitable* if

- for every $\langle \mathfrak{f}, l_{\mathfrak{f}} \rangle \in \boldsymbol{F}$ there is $\langle \mathfrak{g}, l_{\mathfrak{g}} \rangle \in \boldsymbol{G}$ such that the pair $\langle \mathfrak{f}, l_{\mathfrak{f}} \rangle$, $\langle \mathfrak{g}, l_{\mathfrak{g}} \rangle$ is suitable;
- for every $\langle \mathfrak{g}, l_{\mathfrak{g}} \rangle \in \boldsymbol{G}$ there is $\langle \mathfrak{f}, l_{\mathfrak{f}} \rangle \in \boldsymbol{F}$ such that $\langle \mathfrak{f}, l_{\mathfrak{f}} \rangle$, $\langle \mathfrak{g}, l_{\mathfrak{g}} \rangle$ is a suitable pair.

Definition 19 (state function) A *state function* for φ is a function I which associates with each $n \in \mathbb{N}$ a quasistate $I(n) = \langle \boldsymbol{F}_n, \Phi_n \rangle$ for φ.

Definition 20 (quasimodel) A state function I with $I(n) = \langle \boldsymbol{F}_n, \Phi_n \rangle$ is called a *quasimodel* for φ if

- the pair $I(m)$, $I(m+1)$ of quasistates is suitable for every $m \in \mathbb{N}$,
- $\chi \mathcal{U} \psi \in \Phi_n$ iff there exists $k > n$ such that $\psi \in \Phi_k$ and $\chi \in \Phi_l$ for all $l \in (n, k)$ and $n \in \mathbb{N}$.

Say that φ is *satisfied* in the quasimodel I if $\varphi \in \Phi_n$ for some $n \in \mathbb{N}$.

Theorem 21 *A formula $\varphi \in \mathcal{ST}_1$ is satisfied in a tt-model iff it is satisfied in a quasimodel for φ.*

Proof ($\Rightarrow$) By Theorem 14, we may assume that φ is satisfied in a tt-model $\mathfrak{M} = \langle \mathfrak{T}, \mathbb{N}, \mathfrak{a} \rangle$ the underlying topological space $\mathfrak{T}$ of which is determined by a disjoint union $\mathfrak{F} = \langle W, R \rangle$ of forks.

For every $n \in \mathbb{N}$, define an equivalence relation $\sim_n$ on the set of forks in $\mathfrak{F}$ by taking $\mathfrak{f} \sim_n \mathfrak{g}$ iff there is an isomorphism σ from $\mathfrak{f}$ onto $\mathfrak{g}$ such that $x \in \mathfrak{a}(t, n)$ iff $\sigma(x) \in \mathfrak{a}(t, n)$, for all $t \in \mathit{term}(\varphi)$ and all x in $\mathfrak{f}$.

Pick a representative of every $\sim_n$-equivalence class and define $\boldsymbol{F}_n$ to be the set of all selected representatives labelled in accordance with $\mathfrak{a}$. Let Φ_n be the set of all formulas in $\mathit{sub}(\varphi)$ that hold in $\mathfrak{M}$ at moment n. It is easy to check that the state function $I(n) = \langle \boldsymbol{F}_n, \Phi_n \rangle$ is a quasimodel satisfying φ.

($\Leftarrow$) Conversely, suppose φ is satisfied in a quasimodel I for φ. Say that a function r which associates with every $n \in \mathbb{N}$ a labelled fork $r(n)$ in $I(n)$ is a *run* through I if, for every $m \in \mathbb{N}$, the pair of $r(m)$, $r(m+1)$ is suitable. Let $\mathcal{R}$ be the set of all runs through I.

Define a topological temporal model $\mathfrak{M} = \langle \mathfrak{T}, \mathbb{N}, \mathfrak{a} \rangle$ as follows. The topological space $\mathfrak{T}$ is determined by the frame $\mathfrak{F} = \langle W, R \rangle$, where $W = \mathcal{R} \times \{a, b, c\}$ and, for all $x, y \in W$, xRy iff $x = y$ or there exists $r \in \mathcal{R}$ with $x = \langle r, a \rangle$ and $(y = \langle r, b \rangle \vee y = \langle r, c \rangle)$. And the assignment $\mathfrak{a}$ is defined by

$$\mathfrak{a}(X, n) = \{\langle r, d \rangle : X \in l_{r(n)}(d_{r(n)})\}.$$

One can readily check by induction that φ is satisfied in $\mathfrak{M}$. ❑

We are going to prove now that, given an $\mathcal{ST}_1$-formula φ, one can effectively recognise whether there exists a quasimodel satisfying φ. The idea of the proof is to show that a satisfiable φ can always be satisfied in a 'periodical' quasimodel of the form

$$I(0), \ldots, I(k), (I(k+1), \ldots, I(l))^*$$

with effectively bounded k and l. Let us first fix some notation concerning sequences (of arbitrary elements). Given a sequence $s = s(0), s(1), \ldots$ and $i \geq 0$, we denote by $s^{\leq i}$ and $s^{>i}$ the head $s(0), \ldots, s(i)$ and the tail $s(i+1), s(i+2), \ldots$ of s, respectively; $s_1 * s_2$ is the concatenation of sequences s_1 and s_2; $|s|$ denotes the length of s and $s^* = s * s * s * \ldots$

Lemma 22 *Let $I = I(0), I(1), \ldots$ be a quasimodel for φ and $I(n) = I(m)$ for some $n < m$. Then $I_{nm} = I^{\leq n} * I^{>m}$ is also a quasimodel for φ.*

If a subsequence of a quasimodel I for φ is also a quasimodel for φ, then we call it a *subquasimodel* of I. For example, I_{nm} in Lemma 22 is a subquasimodel of I.

Lemma 23 *Every quasimodel I for φ contains a subquasimodel $I_1 * I_2$ such that $|I_1| \leq \sharp(\varphi)$ and each quasistate in I_2 occurs in this sequence infinitely many times.*

Let, as before, $I(n) = \langle \boldsymbol{F}_n, \Phi_n \rangle$. Say that a formula $\psi \mathcal{U} \chi \in \Phi_n$ is *realised in m steps* in I if there exists $l \in (0, m+1)$ such that $\chi \in \Phi_{n+l}$ and $\psi \in \Phi_{n+k}$ for all $k \in (0, l)$.

Lemma 24 *Let $I = I_1 * I_2$ be a quasimodel for φ (with quasistates of the form $\langle \boldsymbol{F}_i, \Phi_i \rangle$, $i \in \mathbb{N}$) satisfying the requirements of Lemma 23 and let $n = |I_1| + 1$. Then I contains a subquasimodel of the form $I_1 * I_0 * I_2^{>l}$, for some $l \geq 0$, such that*

(i) $|I_0| \leq |\mathit{sub}(\varphi)| \cdot \sharp(\varphi) + \sharp(\varphi)$;

(ii) *every formula $\psi \mathcal{U} \chi \in \Phi_n$ is realised in $|I_0|$ steps;*

(iii) $I_0(0) = I_2^{>l}(0)$.

Proof Suppose $\psi \mathcal{U} \chi \in \Phi_n$. Then there exists $m > 0$ such that $\chi \in \Phi_{n+m}$ and $\psi \in \Phi_{n+k}$ for all $k \in (0, m)$. Assume now that $0 < i < j < m$ and $I(n+i) = I(n+j)$. In view of Lemma 22, $I_1 * I_2^{\leq i} * I_2^{>j}$ is a subquasimodel of I. It follows that we can construct a subquasimodel $I_1 * I_2^{\leq 1} * I_3$ of I in which $\psi \mathcal{U} \chi$ is realised in $m_1 \leq \sharp(\varphi)$ steps.

Then we consider another formula $\psi' \mathcal{U} \chi' \in \Phi_n$ and assume that it is realised in $m_2 > m_1$ steps. Using Lemma 22 once again (and deleting repeating quasistates in the interval $I_3(m_1), \ldots, I_3(m_2)$) we select a

subquasimodel $I_1 * I_2^{\leq 1} * I_3^{\leq m_1} * I_4$ of I which realises both $\psi \mathcal{U} \chi$ and $\psi' \mathcal{U} \chi'$ in $2\sharp(\varphi)$ steps.

Having analysed all distinct formulas of the form $\psi \mathcal{U} \chi \in \Phi_n$ we obtain a subquasimodel $I_1 * I_2^{\leq 1} * I'$ of I which realises all these formulas in $|sub(\varphi)| \cdot \sharp(\varphi)$ steps. And $\leq \sharp(\varphi)$ additional quasistates may be required to comply with (iii). ❑

Lemma 25 *Suppose I_1 and I_2 are finite sequences of quasistates for φ of length l_1 and l_2, respectively, and let $I = I_1 * I_2^*$ with $I(n) = \langle \boldsymbol{F}_n, \Phi_n \rangle$. Then I is a quasimodel for φ whenever the following conditions hold:*

1. *for every $i \leq l_1 + l_2$, the pair $\boldsymbol{F}_i$, $\boldsymbol{F}_{i+1}$ is suitable;*
2. *for every $i \leq l_1 + l_2$ and every $\psi \mathcal{U} \chi \in sub(\varphi)$, we have $\psi \mathcal{U} \chi \in \Phi_i$ iff either $\chi \in \Phi_{i+1}$ or $\psi \in \Phi_{i+1}$ and $\psi \mathcal{U} \chi \in \Phi_{i+1}$,*
3. *for every $i \leq l_1 + 1$, all formulas of the form $\psi \mathcal{U} \chi \in \Phi_i$ are realised in $l_1 + l_2 - i$ steps.*

As a consequence of the two preceding lemmas we immediately obtain:

Theorem 26 *An $\mathcal{ST}_1$-formula φ is satisfiable in a topological temporal model iff there are two sequences I_1 and I_2 of quasistates for φ such that $I = I_1 * I_2^*$ satisfies conditions 1–3 of Lemma 25, all quasistates in I_1 are distinct (and so $|I_1| \leq \sharp(\varphi)$),*

$$|I_2| \leq |sub(\varphi)| \cdot \sharp(\varphi) + \sharp(\varphi),$$

and $\varphi \in I(1)$.

This theorem provides us with an EXPSPACE algorithm which is capable of deciding whether a given $\mathcal{ST}_1$-formula φ is satisfiable in a quasimodel for φ. Here is a rough description of such an algorithm; in view of the equality EXPSPACE = NEXPSPACE, it can be non-deterministic. Let $\ell(\varphi)$ be the length of φ.

First we guess $l_1 \leq \sharp(\varphi)$ and $l_2 \leq |sub(\varphi)| \cdot \sharp(\varphi) + \sharp(\varphi)$ and write them in binary using thereby exponential space in $\ell(\varphi)$. Then we guess a set $\boldsymbol{F}_0$ of $\leq 2^{2 \cdot |term(\varphi)|}$ labelled forks, a subset $\Phi_0 \subseteq sub(\varphi)$ containing φ, and check that $\langle \boldsymbol{F}_0, \Phi_0 \rangle$ is a quasistate. In the same way we guess a quasistate $\langle \boldsymbol{F}_1, \Phi_1 \rangle$ (here we do not need the condition $\varphi \in \Phi_1$) and check whether the pair $\langle \boldsymbol{F}_0, \Phi_0 \rangle$, $\langle \boldsymbol{F}_1, \Phi_1 \rangle$ is suitable and whether condition 2 of Lemma 25 is satisfied. After that we remove $\langle \boldsymbol{F}_0, \Phi_0 \rangle$, guess $\langle \boldsymbol{F}_2, \Phi_2 \rangle$, check the pair $\langle \boldsymbol{F}_1, \Phi_1 \rangle$, $\langle \boldsymbol{F}_2, \Phi_2 \rangle$, and so on till we reach $\langle \boldsymbol{F}_{l_1+1}, \Phi_{l_1+1} \rangle$—this quasistate is stored in memory together with the set Σ of all formulas of the form $\chi \mathcal{U} \psi$. We proceed further in the same way as before deleting $\chi \mathcal{U} \psi$ from Σ every time we reach Φ_i containing ψ. If the pair $\langle \boldsymbol{F}_{l_1+l_2}, \Phi_{l_1+l_2} \rangle$, $\langle \boldsymbol{F}_{l_1+1}, \Phi_{l_1+1} \rangle$ is suitable and Σ is empty then φ is satisfiable. This proves Theorem 4 (i).

Suppose now that φ is an $\mathcal{ST}_0$-formula. It follows from the proof of Theorem 14 that φ is satisfiable iff it is satisfied in a quasimodel all quasistates in which contain $\leq c \cdot \ell(\varphi)$ labelled forks, $c = const$. Thus

$$\sharp(\varphi) \leq c \cdot \ell(\varphi) \cdot 2^{2 \cdot |term(\varphi)|} \cdot 2^{|sub(\varphi)|},$$

and so the satisfiability problem for $\mathcal{ST}_0$-formulas is decidable in PSPACE. Recall that the satisfiability problem for pure PTL-formulas is PSPACE-complete (see e.g. (Gabbay *et al.* 1994)). Given such a formula ϕ, we replace every propositional variable p in it with the RCC-8 predicate $\mathsf{EQ}(X^p, Y^p)$ thus obtaining an $\mathcal{ST}_0$-formula ϕ°. It should be clear that ϕ is satisfiable (in a model for PTL) iff ϕ° is satisfiable (in a tt-model). Consequently, the satisfiability problem for $\mathcal{ST}_0$-formulas is PSPACE-complete. This proves Theorem 3.

As to $\mathcal{ST}_1'$-formulas, it is not hard to see that such a formula φ is satisfiable in a tt-model iff it is satisfied in a model $\langle \mathfrak{T}, \mathfrak{L}, \mathfrak{a} \rangle$, where $\mathfrak{L}$ is a strict linear order with $\leq \ell(\varphi)$ points and $\mathfrak{T}$ is determined by a set of $\leq (\ell(\varphi))^2$ forks. This yields a satisfiability checking algorithm which is in NP, and thus proves Theorem 4 (ii).

Let us consider now $\mathcal{ST}_2$-formulas and assume again that we have only one temporal operator $\mathcal{U}$. Suppose $\varphi \in \mathcal{ST}_2$. As before $term(\varphi)$ is the set of region terms occurring in φ; besides region variables it may contain now terms of the form $\Diamond^+ t$, $\Box^+ t$, and $\bigcirc t$.

Definition 27 (run) Suppose that I is a state function for φ. By a *run* r in I we mean a function $r = \langle r_1, r_2, r_3 \rangle$ with domain $\mathbb{N}$ such that, for all $n \in \mathbb{N}$, there exists a fork $\mathfrak{f} = \langle W_\mathfrak{f}, R_\mathfrak{f} \rangle$ underlying some labelled fork in $\boldsymbol{F}_n$ such that $r_1(n)$ is the root of $\mathfrak{f}$ and $r_2(n), r_3(n)$ are its immediate successors, and for all $i \in \{1, 2, 3\}$ and $n \in \mathbb{N}$,

- $\bigcirc t \in l_\mathfrak{f}(r_i(n))$ iff $t \in l_\mathfrak{g}(r_i(n+1))$,
- $\Box^+ t \in l_\mathfrak{f}(r_i(n))$ iff $t \in l_\mathfrak{g}(r_i(m))$, for all $m > n$,
- $\Diamond^+ t \in l_\mathfrak{f}(r_i(n))$ iff there exists $m > n$ such that $t \in l_\mathfrak{g}(r_i(m))$.

Definition 28 (FSA-quasimodel) A pair $\langle I, \mathcal{R} \rangle$ consisting of a state function I with $I(n) = \langle \boldsymbol{F}_n, \Phi_n \rangle$ and a *finite* set of runs $\mathcal{R}$ in I is called an *FSA-quasimodel* for φ if

- $\chi \mathcal{U} \psi \in \Phi_n$ iff there exists $k > n$ such that $\psi \in \Phi_k$ and $\chi \in \Phi_l$ for all $l \in (n, k)$ and $n \in \mathbb{N}$.

- for every $n \in \mathbf{N}$ and every fork $\mathfrak{f}$ underlying a labelled fork in $\boldsymbol{F}_n$, there exists $r \in \mathcal{R}$ such that $W_{\mathfrak{f}} = \{r_1(n), r_2(n), r_3(n)\}$.

Say that φ is *satisfied* in $\langle I, \mathcal{R}\rangle$ if $\varphi \in \Phi_n$ for some $n \in \mathbf{N}$.

Theorem 29 *The following conditions are equivalent:*

1. *A formula $\varphi \in \mathcal{ST}_2$ is satisfied in a tt-model with FSA;*
2. *φ is satisfied in an FSA-quasimodel for φ;*
3. *φ is satisfied in a tt-model based on a finite space.*

Proof (1) $\Rightarrow$ (2). By Theorem 14, we may assume that φ is satisfied in a tt-model $\mathfrak{M} = \langle \mathfrak{T}, \mathbf{N}, \mathfrak{a}\rangle$ with FSA the underlying topological space $\mathfrak{T}$ of which is determined by a disjoint union $\mathfrak{F} = \langle W, R\rangle$ of forks.

For every $n \in \mathbf{N}$, define an equivalence relation $\sim_n$ on the set of forks in $\mathfrak{F}$ by taking $\mathfrak{f} \sim_n \mathfrak{g}$ iff there is an isomorphism σ from $\mathfrak{f}$ onto $\mathfrak{g}$ such that $x \in \mathfrak{a}(t,n)$ iff $\sigma(x) \in \mathfrak{a}(t,n)$, for all $t \in \mathit{term}(\varphi)$ and all x in $\mathfrak{f}$. Observe that, in view of FSA, the set $\{\sim_n : n \in \mathbf{N}\}$ is finite. Pick a representative of every $\sim_n$-equivalence class (such that the representatives coincide whenever $\sim_n = \sim_m$) and define $\boldsymbol{F}_n$ to be the set of all selected representatives labelled in accordance with $\mathfrak{a}$.

Let $\mathfrak{f}$ be a fork in $\mathfrak{F}$. Define $r^{\mathfrak{f}} = \langle r_1^{\mathfrak{f}}, r_2^{\mathfrak{f}}, r_3^{\mathfrak{f}}\rangle$ as follows. For each $n \in \mathbf{N}$, let σ_n be the isomorphism from $\mathfrak{f}$ onto the representative $\mathfrak{f}^n$ of $\mathfrak{f}$ in $\boldsymbol{F}_n$ such that $x \in \mathfrak{a}(t,n)$ iff $\sigma_n(x) \in \mathfrak{a}(t,n)$, for all t and $x \in W_{\mathfrak{f}}$. Put $r_1^{\mathfrak{f}}(n) = \sigma_n(a_{\mathfrak{f}})$, $r_2^{\mathfrak{f}}(n) = \sigma_n(b_{\mathfrak{f}})$, $r_3^{\mathfrak{f}}(n) = \sigma_n(c_{\mathfrak{f}})$. It is not hard to see that all $r^{\mathfrak{f}}$ are runs and that the set

$$\mathcal{R} = \{r^{\mathfrak{f}} : \mathfrak{f} \text{ a fork in } \mathfrak{F}\}$$

is finite. Let Φ_n be the set of all formulas in $\mathit{sub}(\varphi)$ that hold in $\mathfrak{M}$ at moment n. It is easy to check that the pair $\langle I, \mathcal{R}\rangle$ with $I(n) = \langle \boldsymbol{F}_n, \Phi_n\rangle$ is an FSA-quasimodel satisfying φ.

(2) $\Rightarrow$ (3). Suppose φ is satisfied in an FSA-quasimodel $\langle I, \mathcal{R}\rangle$ for φ. Define a tt-model $\mathfrak{M} = \langle \mathfrak{T}, \mathbf{N}, \mathfrak{a}\rangle$ as follows. The topological space $\mathfrak{T}$ is determined by the frame $\mathfrak{F} = \langle W, R\rangle$, where $W = \mathcal{R} \times \{a, b, c\}$ and, for all $x, y \in W$, we have xRy iff $x = y$ or there exists $r \in \mathcal{R}$ such that $x = \langle r, a\rangle$ and $(y = \langle r, b\rangle \vee y = \langle r, c\rangle)$. And the assignment $\mathfrak{a}$ is defined by

$$\mathfrak{a}(X, n) = \{\langle r, d\rangle : X \in l_{r(n)}(d_{r(n)})\}.$$

It is not hard to check by induction that φ is satisfied in $\mathfrak{M}$. Clearly W is finite.

(3) $\Rightarrow$ (1) is trivial. ❑

Definition 30 (suitable pair) Say that an ordered pair $\langle \mathfrak{f}, l_{\mathfrak{f}}\rangle$, $\langle \mathfrak{g}, l_{\mathfrak{g}}\rangle$ of labelled forks for φ is *suitable* if there is an isomorphism σ from $\mathfrak{f}$ onto $\mathfrak{g}$ such that, for all $\bigcirc t$, $\Box^+ t$, and $\Diamond^+ t$ in $\mathit{term}(\varphi)$ and all $x \in W_{\mathfrak{f}}$,

- $\bigcirc t \in l_{\mathfrak{f}}(x)$ iff $t \in l_{\mathfrak{g}}(\sigma(x))$,
- $\Box^+ t \in l_{\mathfrak{f}}(x)$ iff $t, \Box^+ t \in l_{\mathfrak{g}}(\sigma(x))$,
- $\Diamond^+ t \in l_{\mathfrak{f}}(x)$ iff $t \in l_{\mathfrak{g}}(\sigma(x))$ or $\Diamond^+ t \in l_{\mathfrak{g}}(\sigma(x))$.

An ordered pair Φ, Ψ of formula types is suitable if, for every $\psi_1 \mathcal{U} \psi_2$,

$$\psi_1 \mathcal{U} \psi_2 \in \Phi \text{ iff } \psi_2 \in \Psi \text{ or } \psi_1 \mathcal{U} \psi_2 \in \Psi.$$

Theorem 31 *A formula φ is satisfiable in a tt-model with FSA iff there exist two state functions I_1, I_2 of length $l_1 \leq \sharp(\varphi)$ and $l_2 \leq \natural(\varphi) \cdot (2^{4 \cdot |\mathit{term}(\varphi)|} \cdot 3 \cdot |\mathit{term}(\varphi)| + 2^{2 \cdot |\mathit{sub}(\varphi)|} \cdot |\mathit{sub}(\varphi)| + 1)$, respectively, such that for $I = I_1 * I_2^*$ the following conditions hold:*

1. *(a) for $i < l_1 + l_2$, the pair Φ_i, Φ_{i+1} is suitable,*
   (b) the pair $\Phi_{l_1+l_2-1}, \Phi_{l_1}$ is suitable,
   (c) for every $i \leq l_1$ and every $\psi_1 \mathcal{U} \psi_2 \in \Phi_i$, $\psi_1 \mathcal{U} \psi_2$ is realised in $l_1 + l_2 - i$ steps;
2. *for every $i < l_1 + l_2$ and every labelled fork $\mathfrak{h}_i$ in $\boldsymbol{F}_i$ there is a sequence $\mathfrak{h}_0, \ldots, \mathfrak{h}_{l_1+l_2-1}$ such that:*
   (a) $\mathfrak{h}_j \in \boldsymbol{F}_j$, for every $j < l_1 + l_2$,
   (b) the pair $\mathfrak{h}_j, \mathfrak{h}_{j+1}$ is suitable, $j < l_1 + l_2 - 1$,
   (c) $\mathfrak{h}_{l_1+l_2-1}, \mathfrak{h}_{l_1}$ is suitable;
3. *for every $i \leq l_1$ and every labelled fork $\mathfrak{h}_i$ in $\boldsymbol{F}_i$, there is a sequence $\mathfrak{h}_0, \ldots, \mathfrak{h}_{l_1+l_2-1}$ such that:*
   (a) every $\Diamond^+ t$ from $\mathfrak{h}_i$ is realised in $l_1 + l_2 - i$ steps in $\mathfrak{h}_0, \ldots, \mathfrak{h}_{l_1+l_2-1}$,
   (b) $\mathfrak{h}_j \in \boldsymbol{F}_j$, for every $j < l_1 + l_2$,
   (c) the pair $\mathfrak{h}_i, \mathfrak{h}_{i+1}$ is suitable, $i < l_1 + l_2 - 1$,
   (d) $\mathfrak{h}_{l_1+l_2-1}, \mathfrak{h}_{l_1}$ is suitable.

Theorems 7 and 9 follow from Theorems 29 and 31.

6 TEMPORAL MODELS OF EUCLIDEAN SPACE

Although RCC was formulated as a first-order theory that can be interpreted in arbitrary topological spaces, of course the intended models for various applications are one-, two-, or three-dimensional Euclidean spaces, i.e., $\mathbb{R}^n$ for $n = 1, 2, 3$ with the standard interior operator.[2] Renz (1998) showed that for pure RCC-8 formulas satisfiability in arbitrary topological spaces coincides with satisfiability in $\mathbb{R}$, and so in $\mathbb{R}^n$ for any

[2] Cohn (1997) notes, however, that in some applications discrete or even finite topological spaces may be preferable.

$n > 0$; $\mathbb{R}^3$ is enough to realise any set of satisfiable RCC-8 formulas using only connected regions.

Let us observe first that this result of (Renz 1998) cannot be generalised to RCC-8 extended with the operation $\vee$ intended to form unions of regions.

Proposition 32 *There exists a satisfiable* RCC-8 *formula φ with $\vee$ which is not satisfiable in any connected*[3] *topological space. In particular, φ is not satisfiable in $\mathbb{R}^n$ for any $n \geq 1$.*

Proof Take the conjunction φ of the following predicates: $\mathsf{EQ}(X_1 \vee X_2, Y)$, $\mathsf{NTPP}(X_1, Y)$, $\mathsf{NTPP}(X_2, Y)$, $\mathsf{DC}(Y, Z)$. Clearly, φ is satisfied in the topological space consisting of three points and having the identical interior operator. Note now that if φ holds in some topological space, then the region $X_1 \vee X_2$ is closed and included in the interior of Y. On the other hand, it coincides with Y. Hence Y is both closed and open. However, Y is not the whole space because it is disjoint with Z. ❑

Since the operation of forming unions of regions is implicitly available in the language $\mathcal{ST}_2$ (in the form of $\Diamond^+$), we obtain the following:

Proposition 33 *There is an $\mathcal{ST}_2$-formula satisfiable in some tt-model with FSA, but not in a model based on a connected topological space, in particular $\mathbb{R}^n$, for any $n \geq 1$.*

Proof Let φ be the conjunction of the predicates $\mathsf{EQ}(\Diamond^+ X, Y)$, $\mathsf{NTPP}(\bigcirc X, Y)$, $\mathsf{NTPP}(\bigcirc \Diamond^+ X, Y)$, and $\mathsf{DC}(Y, Z)$. As in the proof above, one can show that if φ holds at some moment of time, then Y at that moment must be clopen. ❑

Fortunately, this is not the case for $\mathcal{ST}_1$-formulas.

Theorem 34 *If a set of $\mathcal{ST}_1$-formulas is satisfiable in a tt-model then it is also satisfiable in a model based on $\mathbb{R}^n$, for any $n \geq 1$.*

Proof As follows from the proof of Theorem 14, it suffices to show that if a set Γ of RCC-8 predicates and their negations has a topological model determined by a Kripke model $\mathfrak{M} = \langle \mathfrak{F}, \mathfrak{V} \rangle$ in which $\mathfrak{F}$ is a disjoint union of countably many forks, then Γ has a topological model based on $\mathbb{R}$.

So let us assume that Γ has such a model $\mathfrak{M}$ and $\mathfrak{F}$ is the disjoint union of ω forks $\mathfrak{f}_n$, $n < \omega$, with three points: a_n, the root, and its two successors b_n and c_n.

[3] A space is called *connected* if it is not the union of two disjoint non-empty open sets.

Denote by $\mathcal{X}^n_{ab}$ the set of all region variables X such that

$$\mathfrak{V}(X) \cap \mathfrak{f}_n = \{a_n, b_n\}.$$

Analogously, $\mathcal{X}^n_{ac}$, and $\mathcal{X}^n_{abc}$ are the sets of all region variables X such that

$$\mathfrak{V}(X) \cap \mathfrak{f}_n = \{a_n, c_n\}, \quad \mathfrak{V}(X) \cap \mathfrak{f}_n = \{a_n, b_n, c_n\},$$

respectively. And let $\mathcal{X}^n = \mathcal{X}^n_{ab} \cup \mathcal{X}^n_{ac} \cup \mathcal{X}^n_{abc}$. Define a partial order $\preceq$ on $\mathcal{X}^n$ by taking

$$X \preceq Y \quad \text{iff} \quad \mathfrak{M} \models \forall(X \to Y).$$

For each $n < \omega$, we then choose three maps

- $f^n_{ab} : \mathcal{X}^n_{ab} \mapsto (0, 0.2)$,
- $f^n_{ac} : \mathcal{X}^n_{ac} \mapsto (0, 0.2)$,
- $f^n_{abc} : \mathcal{X}_{abc} \mapsto (0.3, 0.4)$

in such a way that, for every $\bar{e} \in \{ab, ac, abc\}$ and all $X, Y \in \mathcal{X}^n_{\bar{e}}$, $f^n_{\bar{e}}(X) < f^n_{\bar{e}}(Y)$ if $X \preceq Y$, and $f^n_{\bar{e}}(X) = f^n_{\bar{e}}(Y)$ if $X \preceq Y \,\&\, Y \preceq X$. Clearly, such maps exist.

Then we put

- $\mathfrak{a}^n(X) = [n - f^n_{ab}(X), n]$ for every $X \in \mathcal{X}^n_{ab}$,
- $\mathfrak{a}^n(X) = [n, n + f^n_{ac}(X)]$ for every $X \in \mathcal{X}^n_{ac}$,
- $\mathfrak{a}^n(X) = [n - f^n_{abc}(X), n + f^n_{abc}(X)]$ for every $X \in \mathcal{X}^n_{abc}$,
- $\mathfrak{a}^n(X) = \emptyset$ for every $X \notin \mathcal{X}^n$.

Finally, let

$$\mathfrak{a}(X) = \bigcup_{n<\omega} \mathfrak{a}^n(X)$$

for all region variables X. It is a matter of routine now to show that Γ holds in the topological space $\mathbb{R}$ under the assignment $\mathfrak{a}$. We will consider here only two cases.

Suppose $\mathsf{EC}(X, Y) \in \Gamma$. Then there is $n < \omega$ such that

$$a_n \in \mathfrak{V}(X) \cap \mathfrak{V}(Y), \tag{1}$$

but $\bigcup_{n<\omega}\{b_n, c_n\}$ is disjoint with $\mathfrak{V}(X) \cap \mathfrak{V}(Y)$. It follows that $\mathfrak{a}(X)$ and $\mathfrak{a}(Y)$ are externally connected by all n for which (1) holds.

Suppose $\mathsf{NTPP}(X, Y) \in \Gamma$. Then $\mathfrak{a}(X) \subseteq \mathfrak{a}(Y)$ and $\mathfrak{a}(Y) - \mathfrak{a}(X) \neq \emptyset$. It remains to show that $\mathfrak{a}(X)$ is included in the interior of $\mathfrak{a}(Y)$. Let $x \in \mathfrak{a}(X)$. Then $x \in \mathfrak{a}^n(X)$, for some $n < \omega$. It suffices to show that r is in the interior of $\mathfrak{a}^n(Y)$. Three cases are possible:

Case 1: $X \in \mathcal{X}^n_{ab}$. Then $Y \in \mathcal{X}^n_{ab}$ or $Y \in \mathcal{X}^n_{abc}$. If $Y \in \mathcal{X}^n_{ab}$, then x is in the interior of $\mathfrak{a}^n(X)$, since

$f^n_{ab}(X) < f^n_{ab}(Y)$ (recall that $\mathfrak{M} \models \forall(X \to Y)$ but not vice versa). If $Y \in \mathcal{X}^n_{abc}$, then x is in the interior of $\mathfrak{a}^n(Y)$, since the range of f^n_{ab} is contained in $(0, 0.2)$, while the range of f^n_{abc} is contained in $(0.3, 0.4)$.

Case 2: $X \in \mathcal{X}^n_{ac}$. This case is dual to Case 1.

Case 3: $X \in \mathcal{X}^n_{abc}$. Then $Y \in \mathcal{X}^n_{abc}$ (as $\mathfrak{M} \models \forall(X \to Y)$) and $f^n_{abc}(X) < f^n_{abc}(Y)$, from which we deduce that x is in the interior of $\mathfrak{a}^n(Y)$. ❑

7 CONCLUSION

In this paper, we constructed a family of logics intended for qualitative knowledge representation and reasoning about the behaviour of spatial regions in time. We proved that reasoning in these logics is effective and estimated its computational complexity. We also found out the relationship between topological temporal models based on abstract topological spaces and those on Euclidean ones.

The obtained results make the first step in the study of effective spatio-temporal formalisms. Many interesting problems remain open for investigation. Here are some of them, connected with the logics constructed in the paper.

(1) Theorems 4, 7, and 9 establish only upper bounds for the complexity of the satisfiability problem for $\mathcal{ST}^+_i$, $i > 0$. Do they coincide with the lower bounds?

(2) It may be of interest to extend the logics $\mathcal{ST}_i$ with infinitary operators which allow us to construct formulas of the form $\bigwedge_{n\in\mathbf{N}} R(X, \bigcirc^n Y)$ and $\bigvee_{n\in\mathbf{N}} R(X, \bigcirc^n Y)$ to say, for instance, that $\bigvee_{n\in\mathbf{N}} \mathsf{P}(\mathit{Russia}, \bigcirc^n \mathit{EU})$, i.e., eventually the whole Russia as it is today will become part of the EU.

(3) FSA assumes that we can apply $\Box^+$ and $\Diamond^+$ only to region terms that have finitely many possible states. To allow for infinite sets of states, one may consider models that are 'compact' in the sense that regions can change infinitely often and have infinitely many possible states, but there are finitely many maximal and minimal (with respect to $\subseteq$) states starting from each $n \in \mathbf{N}$. We conjecture that the satisfiability problem for $\mathcal{ST}^+_i$-formulas in compact topological temporal models is decidable.

(4) Theorem 15 realises region variables as arbitrary regular closed sets of $\mathbb{R}^n$. What if we require those sets to be connected? (See e.g. (Renz 1998).)

Acknowledgments. We would like to thank B. Bennett, A. Cohn, D. Gabbay, and C. Lutz for stimulating discussions. This work has been partially supported by EPSRC grant number GR/M56807.

References

J. Allen (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983.

B. Bennett (1994). Spatial reasoning with propositional logic. In *Proceedings of the 4th International Conference on Knowledge Representation and Reasoning*, pages 51–62. Morgan Kaufmann, 1994.

B. Bennett (1996). Modal logics for qualitative spatial reasoning. *Journal of the Interest Group on Pure and Applied Logic*, 4:23–45, 1996.

B. Bennett and A. Cohn (1999). Multi-dimensional multi-modal logics as a framework for spatio-temporal reasoning. In *Proceedings of the 'Hot topics in Spatio-temporal reasoning' workshop, IJCAI-99*, Stockholm, 1999.

B. Bennett, A. Cohn, and A. Isli (1997). A logical approach to incorporating qualitative spatial reasoning into GIS. In *Proceedings the International Conference on Spatial Information Theory (COSIT)*, pages 503–504, 1997.

J. Chomicki (1994). Temporal query languages: a survey. In D. Gabbay and H.J. Ohlbach, editors, *Temporal Logic*, volume 827 of *Lecture Notes in Artificial Intelligence*, pages 506–534, Berlin, 1994. Springer-Verlag.

B.L. Clarke (1981). A calculus of individuals based on 'connection'. *Notre Dame Journal of Formal Logic*, 23:204–218, 1981.

B.L. Clarke (1985). Individuals and points. *Notre Dame Journal of Formal Logic*, 26:61–75, 1985.

E. Clementini, J. Sharma, and M.J. Egenhofer (1994). Modeling topological spatial relations: strategies for query processing. *Computers and Graphics*, 18:815–822, 1994.

A. Cohn (1997). Qualitative spatial representation and reasoning techniques. In G. Brewka, C. Habel, and B. Nebel, editors, *KI-97: Advances in Artificial Intelligence*, Lecture Notes in Computer Science, pages 1–30. Springer-Verlag, 1997.

M. J. Egenhofer and D. Mark (1995). Naive geography. In A. Frank and W. Kuhn, editors, *Spatial Information Theory: a theoretical basis for GIS*, volume 988 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, Berlin, 1995.

R. Fagin, J. Halpern, Y. Moses, and M. Vardi (1995).

Reasoning about Knowledge. MIT Press, 1995.

M. Fisher (1994). A survey of concurrent MetateM—the language and its applications. In D. Gabbay and H.J. Ohlbach, editors, *Temporal Logic*, volume 827 of *Lecture Notes in Artificial Intelligence*, pages 480–505, Berlin, 1994. Springer-Verlag.

D. Gabbay, I. Hodkinson, and M. Reynolds (1994). *Temporal Logic*, volume 1. Oxford University Press, 1994.

V. Goranko and S. Passy (1992). Using the universal modality: Gains and questions. *Journal of Logic and Computation*, 2:5–30, 1992.

N.M. Gotts (1996). Using the RCC formalism to describe the topology of spherical regions. Technical Report 96.24, School of Computer Studies, University of Leeds, 1996.

V. Haarslev, C. Lutz, and R. Möller (1999). A description logic with concrete domains and role-forming predicates. *Journal of Logic and Computation*, 9(3):351–384, 1999.

P. Jonsson and T. Drakengren (1997). A complete classification of tractability in RCC-5. *Journal of Artificial Intelligence Research*, 6:211–221, 1997.

Y. Kesten, Z. Manna, H. Mc Guire, and A. Pnueli (1993). A decision algorithm for full propositional temporal logic. In C. Courcoubetis, editor, *Computer Aided Verification, 1993*, pages 97–109, Berlin, 1993. Springer-Verlag.

O. Lemon and I. Pratt (1998). On the incompleteness of modal logics of space: advancing complete modal logics of place. In M. Kracht, M. de Rijke, H. Wansing, and M. Zakharyaschev, editors, *Advances in Modal Logic, Volume 1*, pages 115–132, Stanford, 1998. CSLI Publications.

Z. Manna and A. Pnueli (1992). *The temporal logic of reactive and concurrent systems.* Springer-Verlag, 1992.

Z. Manna and A. Pnueli (1995). *Temporal Verification of Reactive Systems: Safety.* Springer-Verlag, 1995.

P. Muller (1998). A qualitative theory of motion based on spatio-temporal primitives. In A. Cohn, L. Schubert, and S. Shapiro, editors, *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning*, pages 131–142. Morgan Kaufmann, 1998.

W. Nutt (1999). On the translation of qualitative spatial reasoning problems into modal logics. In *Advances in Artificial Intelligence, Proceedings of the 23rd Annual German Conference on Artificial Intelligence.* Springer-Verlag, 1999. To appear.

D. Plaisted (1986). A decision procedure for combinations of propositional temporal logic and other specialized theories. *Journal of Automated Reasoning*, 2:171–190, 1986.

D. Randell, Z. Cui, and A. Cohn (1992). A spatial logic based on regions and connection. In *Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning*, pages 165–176. Morgan Kaufmann, 1992.

J. Renz and B. Nebel (1998). Spatial reasoning with topological information. In C. Freksa, C. Habel, and K. Wender, editors, *Spatial Cognition—An interdisciplinary approach to representation and processing of spatial knowledge*, Lecture Notes in Computer Science, pages 351–372. Springer-Verlag, 1998.

J. Renz and B. Nebel (1999). On the complexity of qualitative spatial reasoning. *Artificial Intelligence*, 108:69–123, 1999.

J. Renz (1998). A canonical model of the region connection calculus. In *Proceedings of the 6th International Conference on Knowledge Representation and Reasoning*, pages 330–341. Morgan Kaufmann, 1998.

J. Renz (1999). Maximal tractable fragments of the region connection calculus: a complete analysis. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence, IJCAI*, pages 448–454. Morgan Kaufman, 1999.

M.H. Stone (1937). Application of the theory of Boolean rings to general topology. *Transactions of the American Mathematical Society*, 41:321–364, 1937.

A. Tarski (1938). Der Aussagenkalkül und die Topologie. *Fundamenta Mathematicae*, 31:103–134, 1938.

J. van Benthem (1996). Temporal logic. In D. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 4*, pages 241–350. Oxford Scientific Publishers, 1996.

F. Wolter and M. Zakharyaschev (1999). Multi-dimensional description logics. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence, IJCAI*, pages 104–109. Morgan Kaufman, 1999.

Spatial Locations via Morpho-Mereology

M. Cristani
Dipartimento Scientifico e Tecnologico
Universitá di Verona, I-37134 Verona, Italy
cristani@sci.univr.it

A.G. Cohn, B. Bennett
School of Computer Studies
University of Leeds, Leeds, LS2 9JT, UK
{agc,brandon}@scs.leeds.ac.uk

Abstract

We present a calculus for representing and reasoning about the location of rigid objects which may move within some region (we will speak of *mobile parts*). The calculus has both a mereological primitive and a morphological one, hence the title of the paper. We present an axiomatisation for congruence, our chosen morphological primitive, define the notion of mobile part, describe a subset of morpho-mereological relations suitable for representing spatial locations, and analyze the computational complexity of this set.

1 Introduction

Developing formalisms for representing and reasoning about qualitative spatial information is now an active research area, both within AI, and within the field of geographical information systems [14]. Much of the effort has been devoted to developing efficient representations for reasoning about topological information [2, 25, 24, 20], although other aspects such as orientation [22, 19], distance [18] and qualitative morphology [12] have also been investigated. Qualitative representations have a natural facility to handle indefinite and imprecise information by abstracting away from metrical details. However, specific formalisms have also been developed to facilitate representing and reasoning with indefinite information. For example, [10] and, independently, [6] have developed extensions of two related formalisms (i.e. [9] and [13] respectively) for representing and reasoning about mereological relations between spatial regions. Common to both these formalisms is the notion of an "egg-yolk"[1] : intuitively, this is a pair of regions, an "egg" and a "yolk", such that the yolk is always a part of the egg; the yolk represents a minimum extension of the indefinite region, whilst the egg represents its maximum extension. This idea has since been extended in various ways [10, 7] but the underlying motivation has remained the same: to represent and reason about the indefiniteness of a region's boundary. An example of an egg-yolk can be found in figure 1.

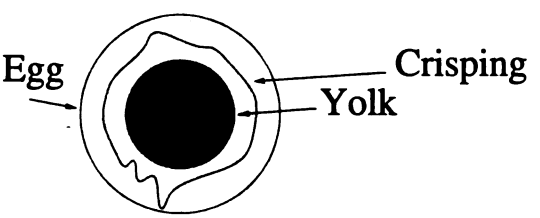


Figure 1: An Egg-Yolk structure

However, there is another notion of indefiniteness which may apply to a region: its location. [8] have proposed a qualitative calculus for representing and reasoning about the position of point-like spatial entities. [27, 26] has proposed a technique for representing and reasoning about the location of spatial regions by locating them with respect to a background region partition (e.g. the States of the USA) using binary relations from a prexisting qualitative topological calculus (the RCC-8 calculus [9]).

In this paper we propose an alternative intepretation of the egg-yolk calculus to represent and reason about the location of objects. However, we shall also make the assumption that the spatial extensions of the objects involved have a fixed morphology. This is motivated by the possible application of the theory to reasoning about the location of actual physical objects, which in general do not tend to change their shape[2].

[1] In fact [6] do not use the term "egg-yolk"; however the concept is essentially the same.

[2] Of course some physical objects do, in particular ani-

We adopt here the following notion of location: an object a is located within a region y *iff* the region x, occupied by a, is a proper part of y. Any egg-yolk pair therefore can be regarded as giving the location of an object (occupying the yolk region) within a hosting region (the egg).

Clearly the location of an object may change over time. However, if the morphology of an object remains constant, then its spatial extensions over time are all congruent to each other.

Our theory will be based on a small number of primitives. Firstly, we assume a mereological primitive, the binary relation of parthood, $\mathsf{P}xy$ — our intuition is that location is primarily based on the notion that the spatial extension of an object is *part* of the spatial extension of another object. From this primitive it is possible to define a set of relations, RCC-5 [21, 1, 20] which consists of five pairwise disjoint and mutually exhaustive relations: EQ , DR , PO , PP , PP^{-1} , representing equality, discreteness, partial overlap, proper parthood and its inverse respectively. Figure 2 depicts the RCC-5 relations. The complexity of determining satisfiability in RCC-5 has been analysed in [20]. Using RCC-5 allows us to represent the notion of the object in question occupying part of another region – we term the object a *mobile part* of its *host* y. In the sequel we shall also have recourse to use certain other concepts which can be defined [1] in terms of $\mathsf{P}xy$: the sum of two regions, $x + y$, and U, the universal region, which we assume here to be unbounded. $\mathsf{O}x, y$ is defined as $\neg\mathsf{DR}x, y$, whilst $\mathsf{Diff}xyz$ is true when z is the region which remains after subtracting y from x.

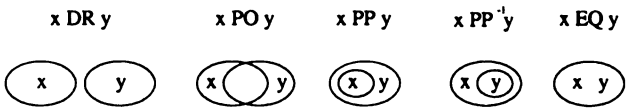


Figure 2: The five basic relations of RCC-5

The second primitive is the binary relation of one spatial region being congruent to another, which we denote by CG. From this, a set of four pairwise disjoint and mutually exhaustive relations can be defined: CG, CGPP, CGPP^{-1} , CNO , representing, congruence, congruent to a proper part and its inverse, and the residual relation in which neither region is congruent to part of the other.

mate ones, but the case of fixed morphology is of interest in any case. Also note that whereas it may be argued that physical objects do not overlap properly, in this paper our intended interpretation is 2D space and projections of 3D objects in 2D space may indeed overlap.

D1) $\mathsf{CGPP}xy \equiv_{def} \exists z\mathsf{PP}zy \wedge \mathsf{CG}xz$
D2) $\mathsf{CGPP}^{-1}xy \equiv_{def} \mathsf{CGPP}yx$
D3) $\mathsf{CNO}xy \equiv_{def} \neg\mathsf{CGPP}xy \wedge \neg\mathsf{CGPP}^{-1}xy \wedge \neg\mathsf{CG}xy$

We term this calculus MC-4 – see figure 3; the complexity of the calculus has been analysed in [12]. Below, we will also make use of CGP (congruent to a part), defined thus:

D4) $\mathsf{CGP}xy \equiv_{def} \exists z\mathsf{P}zy \wedge \mathsf{CG}xz$

In the remainder of the paper first we present an axiomatisation for congruence, our chosen morphological primitive; then we define the notion of *mobile part*, our locational predicate, describe a subset of mereological relations suitable for representing spatial locations, and analyze the computational complexity of this set.

2 Axiomatization of Congruence

In this section we discuss the concept of congruence from the perspective of analyzing the nature of a spatial location.

Given the preexisting axiomatization of RCC-5 [21], we now require an axiomatization of congruence and its relation to RCC-5. Borgo, Guarino and Masolo [5], present a region-based axiomatization of congruence; however, it is inspired by Tarski [29] and is only achieved indirectly by defining an additional predicate, $\mathsf{SPHERE}(x)$ in terms of which congruence is axiomatised[3]; the resulting theory is rather cumbersome. Although we could take this indirect route, we propose the following axiomatisation[4]:

A1) $\mathsf{CG}xx$
A2) $\mathsf{CG}xy \rightarrow \mathsf{CG}yx$
A3) $[\mathsf{CG}xy \wedge \mathsf{CG}yz] \rightarrow \mathsf{CG}xz$
A4) $\mathsf{PP}xy \rightarrow \neg\mathsf{CG}xy$
A5) $\mathsf{P}xU \vee \exists y[\neg\mathsf{CGP}xy \wedge \neg\mathsf{CGP}yx]$
A6) $\mathsf{P}Ux \vee \exists yz[\mathsf{P}yx \wedge \mathsf{P}yz \wedge \neg\mathsf{P}zx \wedge \mathsf{CG}xz]$
A7) $\forall xy[\exists x'[\mathsf{CG}x'x) \wedge \mathsf{P}x'y] \rightarrow \exists y'[\mathsf{CG}y'y \wedge \mathsf{P}xy']]$
A8) $\mathsf{CG}(x_1 + y_1)z \rightarrow$
$\exists(x_2y_2)[\mathsf{CG}x_1x_2 \wedge \mathsf{CG}y_1y_2 \wedge \mathsf{EQ}z(x_2 + y_2)]$

The first three axioms simply specify that CG is an equivalence relation and the fourth that a proper part cannot be congruent to the whole; the remaining axioms establish various key existential properties. It

[3] They also assume a topological primitive of connection in addition to their congruence and mereological primitives, whereas as will be seen below we do not need the topological primitive.

[4] Free variables are assumed to be implicitly universally quantified with maximal scope.

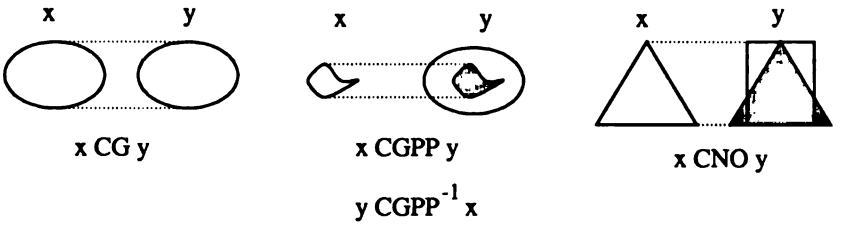


Figure 3: The four basic relations of MC-4

should be noted that we have no guarantee that these axioms do indeed capture every relevant property of congruence expressible within the language. Determining a direct and provably complete axiomatisation remains for future work[5].

The following propositions follow straightforwardly from the above axioms and some are used implicitly in section 4:

P1) $\mathsf{EQ}xy \rightarrow \mathsf{CG}xy$[6]

P2) $\mathsf{CG}xy \rightarrow [\mathsf{EQ}xy \vee \mathsf{DR}xy \vee \mathsf{PO}xy]$

P3) $\mathsf{CGPP}xy \rightarrow [\mathsf{PP}xy \vee \mathsf{DR}xy \vee \mathsf{PO}xy]$

P4) $\mathsf{CGPP}^{-1}xy \rightarrow [\mathsf{PP}^{-1}xy \vee \mathsf{DR}xy \vee \mathsf{PO}xy]$

P5) $\mathsf{CNO}xy \rightarrow [\mathsf{DR}xy \vee \mathsf{PO}xy]$

P6) $\mathsf{EQ}xy \rightarrow \mathsf{CG}xy$

P7) $\mathsf{PP}xy \rightarrow \mathsf{CGPP}xy$

P8) $\mathsf{PP}^{-1}xy \rightarrow \mathsf{CGPP}^{-1}xy$

3 Axiomatisation of Mobile Part

We referred earlier to the notion of a "mobile part". The notion we wish to capture is that of an object x and a hosting region y such that x is a part of y and is free to move everywhere within y; in particular it should be possible for x to "cover" any point in y and for x to move between any two positions whilst remaining entirely within y.

First we will define the notion of being able to place the mobile part anywhere within the hosting region, $\mathsf{CBPA}xy$ (x can be placed anywhere within y):

D5) $\mathsf{CBPA}xy \equiv_{def}$
$\forall z[\mathsf{P}zy \rightarrow \exists w \exists t[\mathsf{P}wz \wedge \mathsf{P}wt \wedge \mathsf{P}ty \wedge \mathsf{CG}tx]]$

We now turn our attention to the other requirement of MP: being "free to move everywhere". To define this is not at all easy. Indeed it is not clear at first sight whether mereology and congruence are sufficiently powerful to achieve this: if x is not "free to move everywhere" within y, then there is some position x_1 in y congruent to x such that x cannot move to another position x_2 also in y and congruent to x such that x remains in y throughout the move. What might stop this happening, is some kind of constriction which blocks x's passage – when this happens x effectively 'plugs' the constriction and the remainder of y is separated into two parts. Being separated into more than one part is a topological notion which is not certainly not definable from mereology alone[30]. However, we have recently managed to define a predicate to test for a region being a sphere using P and CG [4]; combined with a definition of concentricity one can then define the connection relation, $\mathsf{C}(x, y)$ (which is the basis of the mereotopological language RCC-8 [23, 17, 9] and indeed many other related languages [11]) and thus one pieceness.

In [4] we present a logical framework for qualitative reasoning about spatial situations and movements based on the primitive relations of *parthood* and *congruence*. We then show how our formalism can provide qualitative descriptions of changing configurations of regions associated with the movement (translations and rotations) of solid objects in a confining environment. In [3] we use this theory to define linear translations and rotations of rigid objects which may be constrained by some enclosing environment, as in our notion of a mobile proper part. We briefly overview this theory here and show how it can be used to define

[5] In section section 3 a complete axiomatisation of a qualitative geometry based on P and CG is outlined; this theory does not need the axioms for congruence given above because completeness is taken care of by the incorporation of Tarski's geometry axioms. However, ideally one would get rid of all the Tarski geometry axioms by caching them out in terms of properties of CG and P.

[6] We assume here that EQ represents equality of spatial regions and justifies substitution in spatial contexts; otherwise this proposition would need to be an axiom.

the notion of mobile part that we require.

We now give a concise summary of the relevant parts of the theory. A fuller explanation can be found in [4, 3].

First we define the following syntax to refer to the sum of all regions satisfying a given predicate:

D6) $\mathsf{SUM}_x\phi(x) : y \equiv_{def}$
$\forall z[\phi z \rightarrow \mathsf{P}zy] \wedge \neg\exists z[\mathsf{P}zy \wedge \forall w[\phi w \rightarrow \mathsf{DR}wz]]$

We now give a series of definitions where the mereological concepts are combined with a primitive *congruence* relation, CG(x,y). These culminate in the specification of a sphere predicate, S[7]. The proof that the definitions mean that $\mathsf{S}(x)$ holds just in case x is a sphere are given in [4].

D7) $\mathsf{CGOP}xyz \equiv_{def} \exists x'[\mathsf{CG}xx' \wedge \mathsf{O}x'y \wedge \mathsf{O}x'z]$
D8) $x \preceq y \equiv_{def} \forall z_1 z_2[\mathsf{CGOP}yz_1z_2 \rightarrow \mathsf{CGOP}xz_1z_2]$
D9) $x \prec y \equiv_{def} x \preceq y \wedge \neg(y \preceq x)$
D10) $\mathsf{MAXCGOP}x \equiv_{def} \forall y[\mathsf{PP}xy \rightarrow y \prec x]$
D11) $\mathsf{CGOSUM}xy \equiv_{def} \mathsf{isrSUM}_z(\mathsf{CG}zx \wedge \mathsf{O}zx : y$
D12) $\mathsf{S}x \equiv_{def} (\mathsf{MAXCGOP}x \wedge \exists y[\mathsf{CGOSUM}yx])$

Now we define some fundamental geometrical relations among spheres, including betweeness and equidistance:

D13) We use Tarski's [28] definitions of the external diametricity (ED) and concentricity (CONC) relations among spheres (from the primitives P and S).
D14) $\mathsf{B}xyz \equiv_{def} \mathsf{EQ}xy \vee \mathsf{EQ}yz \vee \exists vw[\mathsf{ED}xyv \wedge \mathsf{ED}vwy \wedge \mathsf{ED}y\dot{z}w]$
D15) $\mathsf{EQD}xyzw \equiv_{def}$
$(\mathsf{EQ}xy \wedge \mathsf{EQ}zw) \vee$
$\exists x'y'z'w'[\mathsf{CONC}x'x \wedge \mathsf{CONC}y'y \wedge \mathsf{CONC}z'z \wedge \mathsf{CONC}w'w \wedge \mathsf{ET}x'y' \wedge \mathsf{ET}z'w' \wedge \mathsf{CG}x'z' \wedge \mathsf{CG}y'w']$

In [4] sufficient other definitions and axioms are given so that a proof that this theory gives a complete region based geometry is obtained.

3.1 Auxilliary Definitions

The *bounding sphere* for a region is the smallest sphere of which the region is a part. Since our domain includes unbounded regions, not every region has a bounding sphere. We define

D16) $\mathsf{BS}xx' \equiv_{def} \mathsf{P}xx' \wedge \mathsf{S}x' \wedge \neg\exists y[\mathsf{P}xy \wedge \mathsf{S}y \wedge \mathsf{PP}yx']$

It is very useful to be able to specify that a given pair of regions $\langle a, b\rangle$ is congruent to another pair $\langle a', b'\rangle$. By this we mean that $\mathsf{CG}a, a'$ and $\mathsf{CG}b, b'$ and also that the position of a relative to b is the same as the position of a' relative to b'. If a and b are discrete then this is true just in case $\mathsf{CG}(a+b)(a'+b')$; but in general we have to take care of cases where a and b overlap. Thus we define:

D17) $\mathsf{CG}ab;cd \equiv_{def} (\mathsf{CG}aa' \wedge \mathsf{CG}bb' \wedge \mathsf{CG}(a+b)(a'+b')) \wedge$
$(\mathsf{PP}ab \rightarrow \exists xy[\mathsf{Diff}\ bax \wedge \mathsf{Diff}\ b'a'y \wedge \mathsf{CG}xy]) \wedge$
$(\mathsf{PP}ba \rightarrow \exists xy[\mathsf{Diff}\ abx \wedge \mathsf{Diff}\ a'b'y \wedge \mathsf{CG}xy]) \wedge$
$(\mathsf{PO}ab \rightarrow \exists xyzw[\mathsf{Diff}\ abx \wedge \mathsf{Diff}\ bay \wedge \mathsf{Diff}\ a'b'z \wedge \mathsf{Diff}\ b'a'w \wedge \mathsf{CG}(x+y)(z+w)])$

In order to specify motions of rigid objects we first specify the simple motions of linear translation and rotation about the centre point of some sphere.

3.2 Linear Translation

To specify simple linear motions we define the translation of a region x_1 to the congruent region x_2 along a vector defined by the centre points of two discs d_1 and d_2 as follows (see Fig. 4):

D18) $\mathsf{TAV}x_1x_2d_1d_2 \equiv_{def} \exists d[\mathsf{CONC}d'd_2 \wedge \mathsf{CG}x_1d_1; x_2d]$

We can also define translation part-way along a vector:

D19) $\mathsf{PTAV}x_1x_2d_1d_2 \equiv_{def} \exists d[\mathsf{B}d_1dd_2 \wedge \mathsf{CG}x_1d_1; x_2d]$

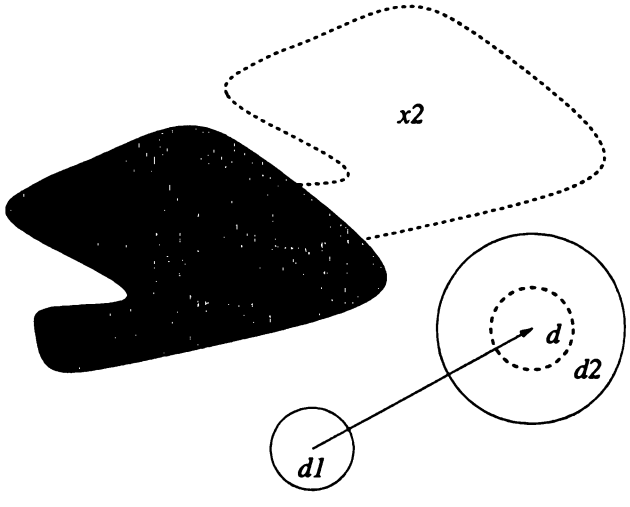


Figure 4: Translation along a vector

3.3 Movement in a Constraining Environment

Our idea is to model a possible movement within an environment as a series of translations and rotations,

[7] In the present paper we are just interested in a 2D world, so a sphere would be better termed a disc, but we we will keep the word sphere for consistency with [4, 3].

during each of which the area occupied by a (rigid) object must always lie within a region of free space. However, in this paper we will only deal with translations; for the formal definitions of rotations see [3].

First we define a *linear translation within* as a translation from x_1 to x_2 along some vector such that all translations part way along the vector lie within a confining region y:

D20) $\mathsf{LTW}x_1x_2y \equiv_{def} \exists d_1d_2[\mathsf{TAV}x_1x_2d_1d_2 \wedge \forall x[\mathsf{PTAV}x_1xd_1d_2 \rightarrow \mathsf{P}xy]]$

We can now define a simple move within, where $\mathsf{RotW}x_1x_2y$ is defined in [3] as being true when x_1 is rotated to x_2 whilst remaining within y:

D21) $\mathsf{SMW}x_1x_2y \equiv_{def} (\mathsf{LTW}x_1x_2y \vee \mathsf{RotW}x_1x_2y)$

We now axiomatise a predicate MoveWithin x_1x_2y to say that a rigid body can move from region x_1 to region x_2, while remaining within region y.

A9) $\mathsf{MoveWithin}\ x_1x_2y \leftrightarrow \mathsf{SMW}x_1x_2y \vee \exists x'[\mathsf{SMW}x_1x'y \wedge \mathsf{MoveWithin}\ x'x_2y]$

Finally we can define the notion of mobile part, $\mathsf{MP}xy$, as being true iff x is part of y, it can be placed anywhere in y and wherever it is placed in y, it can be moved to any other position in y in a continuous fashion and staying inside y:

D22) $\mathsf{MP}xy \equiv_{def} [\mathsf{P}xy \wedge \mathsf{CBPA}xy \wedge \forall x_1\forall x_2[[\mathsf{P}x_1y \wedge \mathsf{P}x_2y \wedge \mathsf{CG}x_1x \wedge \mathsf{CG}x_2x] \rightarrow \mathsf{MoveWithin}\ x_1x_2y]]$

The relation of *Mobile proper part* MPP can be trivially obtained from this.

There are a number of properties of $\mathsf{MP}xy$ which follow from the above definitions; one such property is the fact that y must be a one piece region. The proof of this follows straightforwardly from the definition of $\mathsf{MP}xy$: if y is not one piece, then either there is a piece of y which cannot contain x (but this contradicts the CBPA condition on MP); or y has two pieces, each of which can contain x but then it will be impossible to MoveWithin x from one to the other.

4 Algebraic Analysis

As already mentioned above, the standard interpretation of the egg-yolk model is purely mereological (or topological in [10]) and deals with boundary rather than positional uncertainty. Moreover the shape of the regions involved is able to change arbitrarily (c.f. the indicated crisping compared to the shape of egg and yolk in figure 1). However, here we wish to model a world where the shape and size of objects is constant but it is their position which is indefinite.

Consider for example, a car travelling from Leeds to Sheffield. The car has a given shape and size, which does not change significantly during the trip (provided of course that no accidents occur!). However, the location of the car changes during the trip. If we want to represent the uncertainty deriving from this temporal projection, we need a representation in which shapes and sizes are fixed but uncertainty on the position of the objects is representable.

Locational uncertainty may not depend on temporal projections at all. Consider the notion of "The spatial region occupied by our car in a given parking lot". Again, the morphology of the car is constant (and presumably known to us since it is ours), but we may be uncertain about where we parked it.

In these circumstances the notion of mobile proper part as defined above provides the required framework for location.

Definition 1 *A pair p of the regions y and e, where* $\mathsf{MPP}ye$*, is said to be a "Free Range" Egg-Yolk Configuration (abbreviated* FREYC*). If* $\mathsf{CG}y'y$ *and* $\mathsf{MPP}y'e$ *then y' is sai to be an aternative location for y in the FREYC.*

Thus a FREYC defines a set of possible positions for a given egg-yolk pair, as illustrated in figure 5(a): $\langle y, e\rangle$ is a FREYC (y' and y'' are two of the alternative locations for y). The other cases in figure 5 are not FREYCs because y is not a mobile part of y: in b) y cannot pass from one lobe of e to the other; in c) y is too large to fit within the narrow protrusion in e; and in d) y has an alternative location y' which is trapped by the neck of a recess within e.

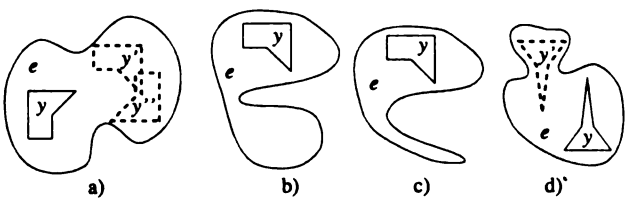


Figure 5: Illustrations of a FREYC (a) and non FREYCs (b,c,d).

If p is a FREYC, then $y(p)$ is the yolk of p and $w(p)$ is the "white" of p, namely $p - y(p)$. Now consider a pair of FREYCs and the set of possible RCC-5 relations[8],

[8]Such a set represents the disjunctive of the constituent

S between the yolks: as they move around, different RCC-5 relations may obtain[9]. How many possible relations are there? In principle there are 2^5 possible general RCC-5 relations, including the empty relation and all the disjunctions of the five base relations. However, given our assumption of fixed morphology, then clearly $\{\mathsf{EQ}, \mathsf{PP}\} \not\subseteq S$ since EQ implies the objects are the same size and PP that one is larger than the other. For similar reasons we can infer that $\{\mathsf{EQ}, \mathsf{PP}^{-1}\} \not\subseteq S$ and $\{\mathsf{PP}, \mathsf{PP}^{-1}\} \not\subseteq S$. After eliminating all relations which include $\{\mathsf{EQ}, \mathsf{PP}\}$, $\{\mathsf{EQ}, \mathsf{PP}^{-1}\}$ or $\{\mathsf{PP}, \mathsf{PP}^{-1}\}$ 15 relations remain. It turns out that by considering the notion of MP(which all FREYCs satsify), further relations can be eliminated, as provided by the following theorems.

Theorem 1 *Given two* FREYC*'s a and b, the mereological relation between $y(a)$ and $y(b)$ is not the union of a relation in* $\{\mathsf{EQ}, \mathsf{PP}, \mathsf{PP}^{-1}\}$ *and the relation* DR.

Proof
First note that if $y(a)$ CNO $y(b)$, then the RCC-5 relation between $y(a)$ and $y(b)$ is not one of EQ , PP or PP^{-1} (this follows from proposition P5 at the end of section 2). Thus the morphological relation between $y(a)$ and $y(b)$ is one of CG, CGPP or CGPP^{-1} . We prove that, if $y(a)$ can only be either equal or disjoint from $y(b)$ then neither a nor b can be a FREYC. The other two cases can be proved analogously.

If the two yolks can be equal, they must be congruent. By hypothesis, there exist regions y_1, y_2, y_3 all mutually DR and all congruent to $y(a)$ and $y(b)$ such that only $y(a)$ can be located at y_1, only $y(b)$ can be located at y_2 and both $y(a)$ and $y(b)$ can be located at y_3. However, because in $\mathcal{F}$ FREYC the yolks are mobile parts of their eggs, there must be a way of moving $y(a)$ from y_1 to y_3 (and similarly $y(b)$ from y_2 to y_3) such that at every intermediate position, the yolk is located within the egg; but it is clear that there must be an intermediate position which overlaps y_3 which contradicts the assumption that the only relationships between the yolks are EQ or DR. A pictorial representation of an example of this situation is presented in figure 6.

Since for no position of y_1 within a_{y_1} and no position of y_2 within a_{y_2} the regions properly overlap the intersection of a_{y_1} and a_{y_2}, then $a_{y_1} \cap a_{y_2} = y_3$.

This means that a_{y_1} and a_{y_2} satisfy the conditions of the claim. The reasoning above can actually be reversed, showing therefore that such conditions are necessarily satisfied by any pair of regions under the claim we are talking about.

However, if these conditions are satisfied, then from y_1 to y_3 there is no path within a_{y_1} which contradicts the interpretation of MP. The same holds for a_{y_2}, proving therefore the claim. ■

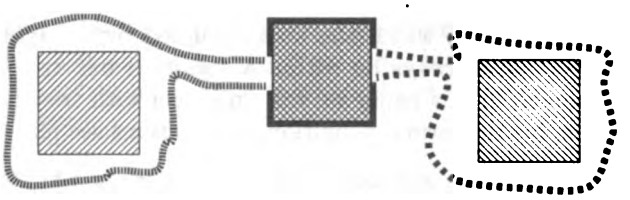


Figure 6: Two Egg-Yolk configurations where the yolks can only be in a EQ, DR relation; however the yolks are not mobile parts of their eggs, so this is an impossible relation.

Theorem 2 *Given two* FREYC*'s a and b, the mereological relation between $y(a)$ and $y(b)$ is not* EQ.

Proof
Sketch. If two regions a and b cannot be 'slopped' to an extent in which they properly overlap, and they are equal, any region containing both which forces this condition is equal to the regions themselves as well. This situation is not compatible with the definition of FREYC's. ■

We now introduce the morpho-mereological configurations of uncertain locations. Consider four regions a, $\widehat{a}$, b and $\widehat{b}$ such that a PP $\widehat{a}$ and b PP $\widehat{b}$. Suppose that we know which MC-4 relations can be established between a and b. The possible configurations depend on:

1. which morphological relation is established between a and b as established in Table 4. In particular:
   - if a CGb, then the possible relations are EQ , $\{\mathsf{EQ}, \mathsf{DR}\}$, $\{\mathsf{EQ}, \mathsf{PO}\}$, $\{\mathsf{EQ}, \mathsf{DR}, \mathsf{PO}\}$, DR , PO , $\{\mathsf{DR}, \mathsf{PO}\}$
   - if a CGPPb, then the possible relations are PP , $\{\mathsf{DR}, \mathsf{PP}\}$, $\{\mathsf{PO}, \mathsf{PP}\}$, $\{\mathsf{PP}, \mathsf{DR}, \mathsf{PO}\}$, DR , PO , $\{\mathsf{DR}, \mathsf{PO}\}$
   - if a CGPP^{-1} b, then the possible relations are PP^{-1} , $\{\mathsf{PP}^{-1}, \mathsf{DR}\}$, $\{\mathsf{PO}, \mathsf{PP}^{-1}\}$, $\{\mathsf{PP}^{-1}, \mathsf{DR}, \mathsf{PO}\}$, DR , PO , $\{\mathsf{DR}, \mathsf{PO}\}$

relations.

[9] It is worth recalling footnote 2 here: even though our objects of interest are solid – a 2D projection of them means they can still 'overlap'.

Rel. in RCC-5	Rel. in MC-4	No. of configs
EQ	CG	0
PP	CGPP	1
PP^{-1}	$CGPP^{-1}$	1
DR	$\top$	4
PO	$\top$	4
{DR, PO}	$\top$	4
{EQ, DR}	CG	0
{EQ, PO}	CG	1
{EQ, DR, PO}	CG	1
{DR, PP}	CGPP	0
{PO, PP}	CGPP	1
{PP, DR, PO}	CGPP	1
$\{PP^{-1}, DR\}$	$CGPP^{-1}$	0
$\{PO, PP^{-1}\}$	$CGPP^{-1}$	1
$\{PP^{-1}, DR, PO\}$	$CGPP^{-1}$	1

Table 1: Summary table: in first column the mereological relations; in second column the corresponding basic morphological relation which holds when the morphology of the "yolk" is fixed (where $\top$ appears it means that the relation is compatible with all the four basic relations of MC-4); in third column the number of morpho-mereological relation pairs derived by theorem 1 and 2.

- if a CNO b, then the possible relations are DR , PO , {DR, PO}

2. which mereological relation is established between $\widehat{a}$ and $\widehat{b}$. In particular, for any relation between $\widehat{a}$ and $\widehat{b}$ other than DR , all the configurations are theoretically allowed, while when $\widehat{a}$ DR $\widehat{b}$, then a DR b.

3. which morphological relation is established between a and $\widehat{a} - \widehat{b}$, b and $\widehat{b} - \widehat{a}$, a, b and $\widehat{a} \cap \widehat{b}$.

The interesting distinctions made under (3) above are when the relations CGP or $\{CGPP^{-1}, CNO\}$ hold between a and $\widehat{a} - \widehat{b}$ or between b and $\widehat{b} - \widehat{a}$. For the case in which both a and b are in the $\{CGPP^{-1}, CNO\}$ relation respectively with $\widehat{a} - \widehat{b}$ and $\widehat{b} - \widehat{a}$, we may distinguish the cases in which they can be kept apart from the case in which they cannot be kept apart.

One more distinction needs to be made in order to obtain a complete classification of possible configurations. Consider the case in which the "eggs" of two FREYC's $\widehat{a}$ and $\widehat{b}$ are in a PP relation. If the yolk a of $\widehat{a}$ is in a CGPP relation to the yolk b of $\widehat{b}$ we have two possible mereological configurations, pictorially represented in Figure 7. In particular, in Figure 7(1) a is big enough to be in a PO relation to b, so the relation between a and b is {PO, PP} , whilst in Figure 7(2) the first yolk is sufficiently small that the relation between a and b is PP .

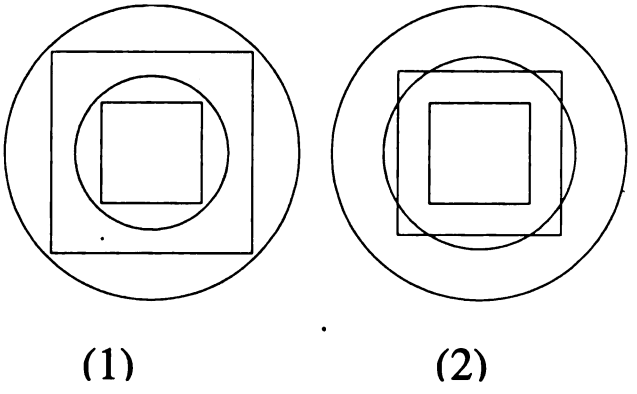


Figure 7: A pictorial representation of the two configurations in which a pair of Egg-Yolks can be when the mereological relation between the eggs is PP and the morphological relation between the yolks is CGPP. The squares represents the yolks and the circle represent the eggs.

Taking account of all these constraints, the number of possible configurations is 60. For the case in which the yolks are congruent to each other the possible cases are the ones listed in Figure 8 with numbers 1-8. The two yolks are depicted with the shaded square shapes, whilst the eggs are the surrounding unshaded squares (the yolk with the dashed boundary belongs to the egg which also has the dashed boundary. These eight cases repeat four times, one for each basic relation of MC-4 established between the yolks.

The cases 9-12 listed in the same Figure represent the 4 cases which arise when one yolk (the triangle) is CGPP to the other one (the square). These four cases repeat (inverted) for the case in which the relation established between the yolks is $CGPP^{-1}$. Finally the four cases repeat twice for the case in which the relation is CNO .

The mereological relations we can establish between the yolks are thus all the relations of RCC-5 which do not contain {EQ, PP} , $\{EQ, PP^{-1}\}$, $\{EQ, PP, PP^{-1}\}$, $\{PP, PP^{-1}\}$ except EQ , {EQ, DR} , {DR, PP} and $\{PP^{-1}, DR\}$. The resulting set, $\mathcal{L}$ is { {EQ, PO}; {EQ, DR, PO}; PP; {PP, PO}; {PP, DR, PO}; PP^{-1}; $\{PP^{-1}, PO\}$; $\{PP^{-1}, DR, PO\}$; PO; DR; {DR, PO} }

A relevant observation we can make about these relations is that they are *continuous* in the sense of Galton [15]. Looking at the Conceptual Neighbourhood Graph of RCC-5 as presented in Figure 9, we indeed note that the eleven relations above can all be ob-

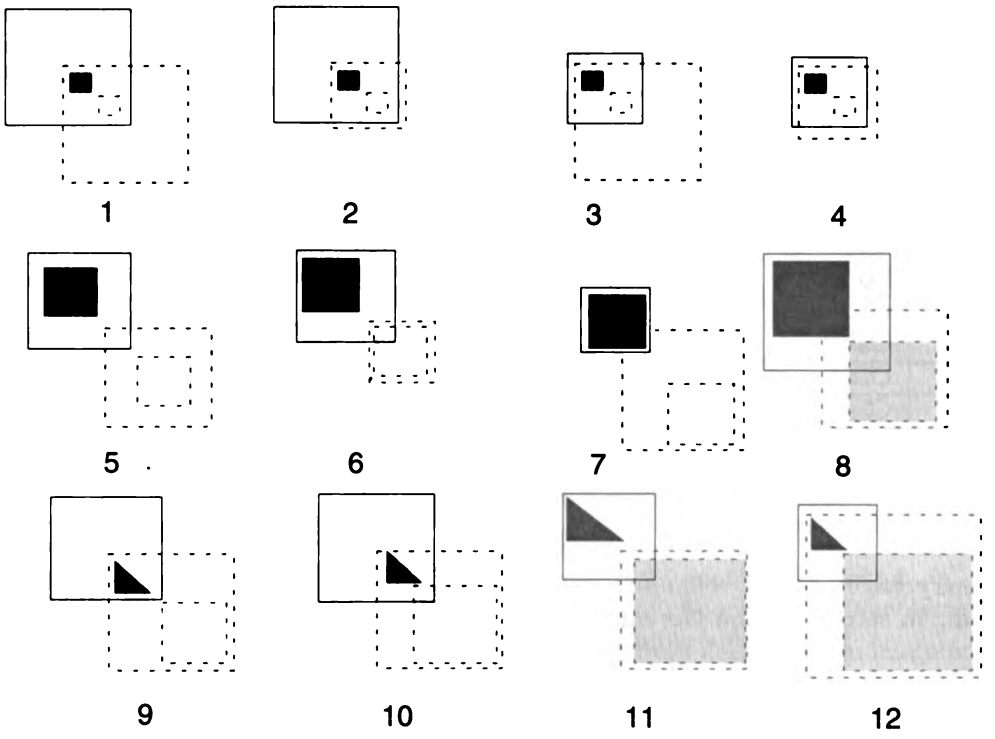


Figure 8: A pictorial representation of basic morphological configurations of Egg-Yolk pairs.

tained by reading labels of vertices through paths of the graph.

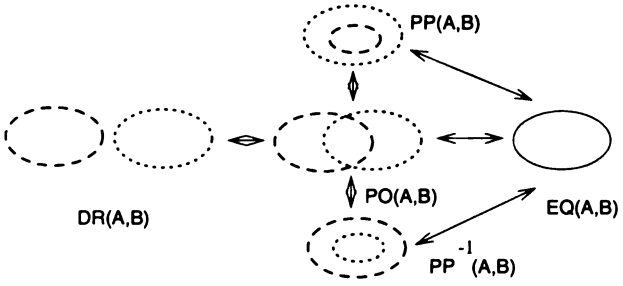


Figure 9: The Conceptual Neighbourhood Graph of RCC-5.

The eleven relations obtained above define distinct groupings of the possible *morpho-mereological basic configurations* we can build.

Definition 2 *Consider two* FREYCs *a and b; we define a* morpho-mereological basic configuration *as a tuple:*

$\langle m_{y(a),y(b)},\ r_{a,b},\ r_{y(a),y(b)},\ m_{y(a),a-b},\ m_{y(b),b-a},\ m_{y(a),a\cap b},\ m_{y(b),a\cap b},\ \alpha,\ \beta\rangle$

where:

- $r_{y(a),y(b)}$ *is the mereological basic relation between the yolk of a and the yolk of b;*
- $r_{y(a),b}$ *is the mereological basic relation between the yolk of a and b;*
- $r_{a,y(b)}$ *is the mereological basic relation between a and the yolk of b;*
- $r_{a,b}$ *is the mereological basic relation between a and b;*
- $m_{y(a),y(b)}$ *is the morphological basic relation between the yolk of a and the yolk of b;*
- $m_{y(a),a-b}$ *is the morphological basic relation between the yolk of a and* $a - b$*;*
- $m_{y(b),b-a}$ *is the morphological basic relation between the yolk of b and* $b - a$*;*
- $m_{y(a),a\cap b}$ *is the morphological basic relation between the yolk of a and* $a \cap b$*;*
- $m_{y(b),a\cap b}$ *is the morphological basic relation between the yolk of b and* $a \cap b$*;*
- α *is the Boolean condition answering the question "Can a be kept apart from b?";*
- β *is the Boolean condition answering the question "Is a small enough to always be a proper part of b?"*

The possible theoretical basic configurations are therefore $5 \cdot 5 \cdot 5 \cdot 5 \cdot 4 \cdot 4 \cdot 4 \cdot 4 \cdot 4 \cdot 2 \cdot 2 = 2,560,000$. Such a large number of configurations clearly cannot be hand analyzed. A machine assisted analysis gives 46 configurations for the mereological part, as was already

known by the original egg-yolk analysis performed in [10] and 190 morphological configurations for the morphological part, reducing the number of actual possible configurations to $46 \cdot 190 = 8740$.

An analysis of these configurations yields 1778 consistent configurations (namely configurations in which the morhpological and the mereological part are consistent with each other and compatible with the "apart" and "only part" conditions). Among these 1778 configurations, we compute three different hierarchical groupings:

- First of all we group all the relations which present the same morphological configuration, and obtain thus 266 groups.
- We then group these 266 configurations into 60 groups in which the mereological relation between the yolk is the same, which correspond to the 60 morpho-mereological configurations we described above.
- Finally we group the 60 configurations into 11 groups, each of which correspond to the same morphological relation between the yolks and the same mereological configuration of the two eggs. The 11 groups obtained this way exactly correspond to the 11 mereological relations introduced by means of the theoretical analysis.

The 11 relations we individuated this way do not form a closed set with respect to the composition, intersection, and converse operation: the closure of $\mathcal{L}$ under composition, intersection and converse, $\widehat{\mathcal{L}}$ is formed by the 11 relations plus three more relations: $\top$, $\bot$ and the relation $\{\mathsf{EQ}, \mathsf{PP}, \mathsf{PP}^{-1}, \mathsf{PO}\}$ (the negation of DR).

However, the set of disjunctive locational relations in RCC-5, namely the set of all unions of the 11 mereological configurations in $\mathcal{L}$, which we denote $\mathcal{L}^{\cup} = \{\bigcup \alpha : \alpha \in 2^{\mathcal{L}}\}$, is closed under composition, intersection and converse. Note that this differs from the situation in the original egg-yolk model [10] where the 13 basic configurations when closed under union is not then closed under composition, intersection and reversal.

$\mathcal{L}^{\cup}$ contains 24 relations. The excluded relations are EQ , $\{\mathsf{EQ}, \mathsf{DR}\}$, $\{\mathsf{DR}, \mathsf{PP}\}$, $\{\mathsf{PP}^{-1}, \mathsf{DR}\}$, $\{\mathsf{EQ}, \mathsf{DR}, \mathsf{PP}\}$, $\{\mathsf{EQ}, \mathsf{DR}, \mathsf{PP}^{-1}\}$, $\{\mathsf{PP}, \mathsf{DR}, \mathsf{PP}^{-1}\}$ and $\{\mathsf{EQ}, \mathsf{PO}, \mathsf{PP}, \mathsf{DR}^{-1}\}$.

The result above is summarized in the next theorem, whose proof is a trivial consequence of the above reasoning.

Theorem 3 *The closure under union of $\mathcal{L}$, $\mathcal{L}^{\cup}$ is a subalgebra of RCC-5.*

The main problem we can now investigate is the computational complexity of reasoning about the relations in the set $\mathcal{L}$. Comparing the set with the four maximal tractable subclasses of RCC-5 described in [20] we can easily note that $\mathcal{L}^{\cup}$ is a proper subset of the maximal tractable subclass which [25] had previously individuated and labelled as $\widehat{\mathcal{H}_5}$.

The immediate consequence of the above observation is the following theorem, where the notation RSAT(X) has to be read as *the satisfiability problem in the subset X*.

Theorem 4 *RSAT($\mathcal{L}^{\cup}$) is polynomial.*

Note that there is no analogous theorem for the 13 original egg-yolk mereological relations: indeed, in that set the satisfiability is not a polynomially solvable task.

This computational result is also of interest because, in the face of the intractability of the RCC-5 and MC-4 algebras, $\mathcal{L}^{\cup}$ provides a "semantical" subalgebra which is polynomially solvable, whilst the maximal subsets of RCC-5 individuated by Jonsson and Drakengren [20] and Renz [25] provide only syntactical ones. This suggests that one might investigate semantically motivated tractable subalgebras of other qualitative spatial languages.

5 Conclusions and further work

We have analysed the problem of representing and reasoning about *locations* of spatial regions in a qualitative way. The analysis has covered three aspects:

- we postulated a set of axioms for congruence expressed in a mereological theory;
- we outlined a mereo-morphological theory within which the notion of a *mobile part* can be defined; based on the notion of mobile part we obtain the notion of *free range egg-yolk*, a new interpretation of the egg-yolk model, a model for spatial uncertainty which previously had only been applied to indefinite boundaries.
- we described the *algebraic* morpho-mereological configurations of spatial locations, and thus individuated a subset of mereological relations in RCC-5 which are suitable of representing the re-

lations between the yolks of a two Free Range Egg-Yolk pairs[10];

- we obtained a tractability result relative to the subset of mereological relations mentioned above.

There are several ways in which this research may be taken further; for example, we could extend the analysis to using RCC-8 rather than RCC-5 or we could combine this model of the uncertainty of spatial location with the previous work egg-yolk model of indeterminate boundaries.

6 Acknowledgements

The work of the first author was funded by the National Project, MURST ex 40% "Metodologie e tecnologie per la gestione di dati e processi su reti internet ed intranet" (Methods and technologies for data and process management on internet and intranet) directed by L. Tanca. The financial assistance of the EPSRC under grant GR/M56807 is also gratefully acknowledged as are the useful discussions with and comments from Paolo Torrini, Shyamanta Hazarika, Laure Vieu and Philippe Muller.

References

[1] B. Bennett. Modal logics for qualitative spatial reasoning. *Bulletin of the Interest Group in Pure and Applied Logic (IGPL)*, 4(1):23–45, 1996. WWW address `ftp://ftp.mpi-sb.mpg.de/pub/igpl/Journal/V4-1/index.html`.

[2] B. Bennett. Determining consistency of topological relations. *Constraints*, 3(2&3):213–225, June 1998.

[3] Brandon Bennett, Anthony G Cohn, Paolo Torrini, and Shyamanta M Hazarika. Describing rigid body motions in a qualitative theory of spatial regions. School of Computer Studies, University of Leeds, 2000.

[4] Brandon Bennett, Anthony G Cohn, Paolo Torrini, and Shyamanta M Hazarika. A foundation for region-based qualitative geometry. School of Computer Studies, University of Leeds, 2000.

[10] The style of the analysis, in which morphological constraints are translated into restrictions on the mereological theory, bears a resemblance to the approach taken to combining distance and topology in [16] where constraints arising from distance induce restrictions on the topological theory.

[5] S. Borgo, N. Guarino, and C. Masolo. A Pointless Theory of Space Based On Strong Connection and Congruence. In L. Carlucci Aiello and J. Doyle, editors, *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR-96)*. Morgan Kaufmann, San Francisco, CA, USA, 1996.

[6] E Clementini and P Di Felice. An algebraic model for spatial objects with undetermined boundaries. In P Burrough and A M Frank, editors, *Proceedings, GISDATA Specialist Meeting on Geographical Entities with Undetermined Boundaries,*. Taylor Francis, 1996.

[7] E Clementini and P Di Felice. Approximate topological relations. *International Journal of Approximate Reasoning*, 16:173–204, 1997.

[8] E Clementini, P Di Felice, and D Hernández. Qualitative representation of positional information. *Artificial Intelligence*, 1997.

[9] A G Cohn, B Bennett, J Gooday, and N Gotts. RCC: a calculus for region based qualitative spatial reasoning. *GeoInformatica*, 1:275–316, 1997.

[10] A G Cohn and N M Gotts. A mereological approach to representing spatial vagueness. In J Doyle L C Aiello and S Shapiro, editors, *Principles of Knowledge Representation and Reasoning, Proc. 5th Conference*, pages 230–241. Morgan Kaufmann, 1996.

[11] A G Cohn and A C Varzi. Connection relations in mereotopology. In H Prade, editor, *Proc. 13th European Conf. on AI (ECAI)*, pages 150–154. J Wiley and Sons, 1998.

[12] M. Cristani. The complexity of reasoning about spatial congruence. *Journal of Artificial Intelligence Research*, 11:361–390, 1999.

[13] M J Egenhofer and R D Franzosa. On the equivalence of topological relations. *International Journal of Geographical Information Systems*, 9(2):133–152, 1995.

[14] Cohn A. G. Qualitative spatial representations. In *Proc. IJCAI-99 Workshop Adaptive Spatial Representations of Dynamic Environments*, 1999.

[15] Antony P Galton. Space, time and movement. In Olivero Stock, editor, *Spatial and Temporal Reasoning*, chapter 10, pages 321–352. Kluwer, Dordrecht, 1997.

[16] A. Gerevini and J. Renz. Combining topological and qualitative size constraints for spatial reasoning. In *Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming (CP'98)*. Springer-Verlag, October 1998.

[17] N M Gotts, J M Gooday, and A G Cohn. A connection based approach to common-sense topological description and reasoning. *The Monist*, 79(1):51–75, 1996.

[18] D. Hernández, E. Clementini, and P. Di Felice. Qualitative distances. In Andrew U. Frank and Werner Kuhn, editors, *Spatial Information Theory: A Theoretical Basis for GIS*, Lecture Notes in Computer Science No. 988, pages 45–58, Semmering, Austria, sep 1995. COSIT'95, Springer-Verlag.

[19] A. Isli and A. G. Cohn. An algebra for cyclic ordering of 2d orientations. In *Proceedings of the 15th American Conference on Artificial Intelligence (AAAI-98)*, pages 643–649, Madison, WI, 1998. AAAI/MIT Press.

[20] P. Jonsson and T. Drakengren. A Complete Classification of Tractability in RCC-5. *Journal of Artificial Intelligence Research*, 6:211–221, 1997. `http://www.cs.washington.edu/research/jair/volume6/jonsson97a.ps`.

[21] F Lehmann and A G Cohn. The EGG/YOLK reliability hierarchy: Semantic data integration using sorts with prototypes. In *Proc. Conf. on Information Knowledge Management*, pages 272–279. ACM Press, 1994.

[22] G Ligozat. Reasoning about cardinal directions. *Journal of Visual Languages and Computing*, 9:23–44, 1998.

[23] D A Randell, Z Cui, and A G Cohn. A spatial logic based on regions and connection. In *Proc. 3rd Int. Conf. on Knowledge Representation and Reasoning*, pages 165–176, San Mateo, 1992. Morgan Kaufmann.

[24] J. Renz. Maximal Tractable Fragments of the Region Connection Calculus: A Complete Analysis. In *Proccedings of the 17th International Conference on Artificial Intelligence (IJCAI 99)*, 1999.

[25] J. Renz and B. Nebel. On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the Region Connection Calculus. *Artificial Intelligence*, 108(1-2):69–123, 1999.

[26] Bittner T. A qualitative coordinate language of location of figures within the ground. In Hirtle S.C. and Frank A.U., editors, *Spatial Information Theory – A Theoretical Basis for GIS (COSIT-97)*, volume 1329 of *Lecture Notes in Computer Science*, pages 223–240, Berlin-Heidelberg, 1997. Springer-Verlag.

[27] Bittner T. and Stell J. A boundary-sensitive approach to qualitative location. *Annals of Mathematics and Artificial Intelligence*, to appear.

[28] A. Tarski. Les fondaments de la géométrie des corps. *Ksiega Pamiatkowa Pierwszego Polskiego Zjazdu Matematycznego*, pages 29–33, 1929. A suplement to *Annales de la Société Polonaise de Mathématique*. English translation, 'Foundations of the Geometry of Solids', in A. Tarski, *Logic, Semantics, Metamathematics*, Oxford Clarendon Press, 1956.

[29] A. Tarski. Foundations of the geoemtry of solids. In J. Corcoran, editor, *Logic, semantics, metamathematics*, pages 24–30. Oxford University Press, Oxford, UK, 1956.

[30] A N Whitehead. *Process and Reality*. The MacMillan Company, New York, 1929. Corrected edition published in 1978 by Macmillan.

Continuous Motion in Discrete Space

Antony Galton
School of Engineering and Computer Science
University of Exeter
Exeter EX4 4PT, UK

Abstract

A number of situations arise in the context of knowledge representation where some notion of continuity is desired within a framework that is itself discrete. We survey some varieties of discrete space that have been proposed, and show that they can all be described as instances of a general notion of closure space, of which topological spaces are a specialised sub-class. We extend the usual topological definition of continuity in the obvious way to general closure spaces, and investigate the possible types of continuous motion that arise when both time and space are represented as closure spaces. In so doing we draw some important connections with existing work on spatio-temporal representations.

1 Introduction

The terms "discrete" and "continuous" are generally regarded as contradictory, so the notion of continuous motion in discrete space may seem paradoxical. Nonetheless, there are a number of situations arising in the context of knowledge representation where some notion of continuity is desired within a framework that is itself discrete.

Consider changes in digital images, for example on a computer screen. The steady motion of a white dot across the screen, one pixel at a time, is qualitatively different from the abrupt motion of a dot which disappears from a location at the left of the screen and reappears instantaneously at the right. We may want to describe the former movement as continuous, the latter as discontinuous, even though at one level of analysis both are "really" discontinuous.

Systems of qualitative descriptors, such as the relations between spatial regions studied in the Regional Connection Calculus (RCC) of (Randell et al. 1992), also "carve up" a continuous space into discrete chunks. Movement in the underlying space gives rise to movement amongst the chunks of the discrete representation, and again we can ask what such movements will look like when the underlying movement is either continuous or discontinuous. A partial answer to this is provided by the *conceptual neighbourhood diagram* (Freksa 1992) that now routinely accompanies expositions of qualitative systems of this kind. The partial answer is made fuller in the theory of dominance spaces of (Galton 1997*a*, Galton 1997*b*), but that theory was somewhat *ad hoc* and needs to be integrated into a more general theory of continuity for discrete spaces.

These discrete spaces arise from continuous spaces by a "chunking" process (essentially, quotienting out an equivalence relation), but it is also worth studying discrete spaces in their own right. A simple example is the game of chess, in which both space and time are conceptually discrete, even though their physical realisations are as continuous as anything else in this world. The term "quasi-continuous" has been applied to "one-square-at-a-time" motions such as that of the King, in contrast to the Queen which can cover many squares in a single move (Galton 1997*b*). The computer screen provides another example, where not everything that appears should be conceived of as the digital image of some underlying continuous object, and yet once again we might wish to constrain certain changes to be in come sense continuous (e.g., so as to avoid confusing the user with abrupt shifts in what they see).

There is thus a general need to characterise the different kinds of discrete representations and the spaces they presuppose, and to determine the characteristics of different kinds of motion within such spaces. Although continuity is normally regarded as a topological notion, in this paper we will adopt a more general

characterisation of the kinds of space within which we shall work, which will turn out to be general enough to encompass all of the forms of discrete space mentioned above and to provide appropriate definitions of continuity for those spaces.

2 A framework for representing movement

As stated by (Galton 1997*b*), a theory of movement must encompass a representation of the *space* in which the movement occurs, a representation of the *time* within which movements occur, representations of the *objects* which move, and a notion of *position*, by which each object is assigned a spatial location at each time during the period of its existence. Only once these ingredients are in place can we begin to *classify* different kinds of movement, and in particular to investigate notions such as continuity.

These remarks are neutral as between discrete and continuous representations of space and time. In this paper we concentrate on discrete space; time may be either discrete or continuous. We shall not have much to say about objects and their positions: we shall take the view that a position for an object is a region in space, where a region is characterised, in discrete space, as some aggregate of the atomic elements out of which the space is composed. We shall have nothing to say about objects over and above their positions: for our present purposes an object just *is* a temporally indexed series of positions. We recognise that for many other purposes it is essential to probe more deeply into just what is meant by an object.

3 Topological characterisation of continuity

Within the mathematical tradition it is usual to characterise continuity in topological terms. Recall the standard definition, that a function f from one topological space T_1 to another topological space T_2 is continuous so long as, whenever X is open in T_2 then its inverse image $f^{-1}(X)$ is open in T_1. Equivalently, for any set X in T_1, $f(cl(X)) \subseteq cl(f(X))$ (where cl is the topological closure operation).

This enables us to characterise *motion* as continuous, as follows. The motion of an object a can be specified by a function $pos(a) : T \rightarrow P$, where T is the set of times and P is the set of possible positions of a. We assume that both P and T are topological spaces. Then the motion of point object a is continuous so long as the function $pos(a)$ is continuous.

For point objects, P is just the set S of spatial points (e.g., $\mathbb{R}^3$). Most objects are not points, however, and in such cases we have to decide how to represent a possible position for the object. Possible positions of objects are generally regarded as *regions*, and it remains to characterise regions in terms of the representation of space. If space is represented as a set of points then regions are typically regarded as sets of points — usually restricted in some way, e.g., open sets, regular open sets, closed sets, regular closed sets, connected open sets, etc. In order to characterise motion as continuous, we must specify a topology on the set of regions. Emphatically, this is not the same as specifying a topology on the set of points, and there are many ways of deriving a topology on regions from a topology on points.

A standard way of doing this is by defining a *metric* on the space of possible positions. This will normally be defined in terms of a pre-existing metric on S. A widely used example is the *Hausdorff metric*, by which the distance between two point-sets is the least d such that every point in either region is within a distance d of a point in the other. A topology on P is derived from a metric $\Delta \subseteq P \times P$ in the standard way, using the open spherical neighbourhoods of the form $B(p_0; \epsilon) = \{p \in P \mid \Delta(p, p_0) < \epsilon\}$ as a basis. Having got the metric, it is not necessary to explicitly formulate the topology: it is implicit in the standard "ϵ, δ" characterisation of continuity directly in terms of the metric. (See (Galton 1997*a*) for a discussion of this and other metrics in relation to the problem of characterising continuous spatial change).

In this paper we shall look at ways of topologising the space of possible positions when the underlying space of points (or spatial atoms) is discrete. But we shall also consider characterisations of continuity which differ from the topological, invoking *pre-topological* notions instead.

4 Varieties of discrete space

In very general terms, the difference between a discrete space and a continuous space is that in the former type of space each primitive element has a set of "nearest neighbours", elements which are in some sense "closer" to it than any other elements. A formal model has to have some way of making this rather vague characterisation more precise. There appear to be essentially just two ways of doing this, which characterise "nearest neighbourhood" as respectively *adjacency* and *incidence*. The former notion leads to approaches based on Graph Theory, the latter to approaches based on Topology.

In a naïve version of the graph-theoretic approach, a space is defined as a set S of primitive elements (e.g., "pixels") together with an irreflexive, symmetric relation $A \subset S \times S$. When xAy holds, we say that x is adjacent to y. There is much of interest to be gained from adopting this kind of approach (Galton 1999, Stell & Worboys 1997), but when it was investigated in the 70s and 80s in the context of analysing digital images on computer screens, it was found to fall foul of some debilitating flaws, the so-called "connectivity paradoxes" (Kong & Rosenfeld 1989).

The paradoxes arose in connection with rectangular two-dimensional pixel-arrays. Modelling an infinite computer screen as $\mathbb{Z} \times \mathbb{Z}$, there are two natural choices for an adjacency relation on the pixels: *4-adjacency* (orthogonal adjacency), by which each pixel (x, y) is considered to be adjacent to the four pixels $(x, y \pm 1), (x \pm 1, y)$ and to no others, and *8-adjacency* (orthodiagonal adjacency), by which in addition each pixel is considered to be adjacent to the four pixels $(x \pm 1, y \pm 1)$. We shall use A_4 and A_8 to denote these adjacency relations. The paradoxes arise when we look for an analogue to the Jordan Curve Theorem. In the continuous space $\mathbb{R} \times \mathbb{R}$, a Jordan Curve is the image of a continuous injective mapping $f : [0, 1) \to \mathbb{R}^2$ such that $\lim_{x \to 1} f(x) = f(0)$. The theorem states that any Jordan Curve C divides $\mathbb{R}^2 \setminus C$ into exactly two connected components (the "inside" and the "outside").

In $\mathbb{Z}^2$, the analogue of a Jordan Curve is a sequence $p_1, p_2, \ldots, p_n$ of distinct primitive elements such that $p_i A_j p_k$ if and only if $|i - k| = 1 \mod n$ (where $j = 4$ or 8). According as $j = 4$ or 8 we may speak of a 4-curve or an 8-curve. Likewise, we shall speak of 4-components or 8-components, according as the connectedness is defined in terms of 4-adjacency or 8-adjacency. In Figure 1, adapted from (Kong et al. 1991), the left-hand illustration shows a set of black pixels which (i) forms a Jordan 8-curve but does not divide its complement into two 8-disconnected components, and (ii) is 4-disconnected but manages to divide its complement into two 4-components. The right-hand illustration shows a set of pixels which (i) forms a Jordan 4-curve which divides its complement into *three* 4-components and (ii) divides its complement into two 8-components but is not a Jordan 8-curve.

Since the Jordan Curve Theorem is regarded as fundamental for reasoning about figures in the plane, approaches which model the digital plane in terms of a single adjacency relation were regarded as deeply flawed. To overcome this problem, Rosenfeld proposed using different adjacencies for a set of pixels and its complement. This makes sense if one can

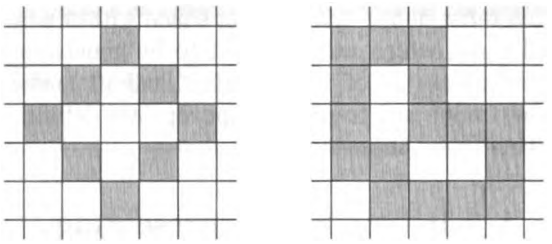


Figure 1: *The connectivity paradoxes.*

systematically distinguish between "foreground" and "background": conventionally, "foreground" regions are represented as sets of black pixels, and "background" regions, their complements, as sets of white pixels. We can either use 8-adjacency for black pixels and 4-adjacency for white, or the other way round; in either case, adjacency between a black pixel and a white pixel follows the rule for white pixels. The left-hand illustration of Figure 1 can now be regarded as a Jordan 8-curve dividing its complement into two 4-components, or as a non-Jordan 4-curve leaving its complement 8-connected, both viewpoints being in accordance with the Jordan Curve Theorem; and similar remarks apply to the right-hand illustration. This approach to digital space was extensively developed by Rosenfeld during the 1970s. It was used for defining image-processing operations such as thinning and segmentation, with particular applications to pattern recognition. See (Kong & Rosenfeld 1989) for a survey.

The adjacency-based approach is not truly topological, even though it makes use of notions such as connectivity which are generally regarded as topological in nature. But there is no concept here of regions being open or closed, nor of the topological notions of interior, boundary or closure — although one can define *border* and *interior* points in terms of adjacency.

This approach is difficult to apply if the distinction between foreground and background is not rigorously enforceable. In Figure 2 (adapted from (Kovalevsky 1989)), for example, we see a white "V" on a black background and a black "V" on a white background. Whether we regard white or black as the foreground colour, and whether we choose 4-adjacency for the background and 8-adjacency for the foreground, or vice versa, it is impossible simultaneously to assign the same connectivity to both figures. And yet on the computer screen it is not uncommon to find both white text on a black background and black text on a white background at the same time.

Problems of this kind led authors such as Kovalevsky to adopt a completely different approach to the for-

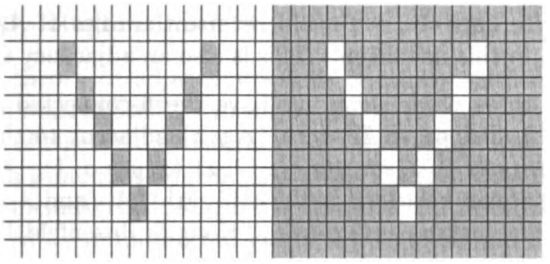


Figure 2: *Interchange of foreground and background.*

mal modelling of discrete space, a topological approach based on the notion of incidence. The key notion here is that of *bounding*. The idea is that in modelling the digital plane, for example, as well as the pixels which overtly appear in an image, one supplements the formal ontology with lower-dimensional entities which form the boundaries between pixels: edges and vertices (sometimes called "linels" and "pointels"). In a rectangular array, a pixel is considered to be bounded by four primitive elements called edges, each of which is in turn bounded by two primitive elements called vertices. The bounding relation is assumed to be irreflexive, asymmetric and transitive, i.e., a strict partial order on the set of primitive elements. Thus a pixel is also bounded by the vertices bounding the edges which bound it. On this basis it is possible to specify a topology on the set of primitive elements. Each element has a *minimal neighbourhood* consisting of the element itself together with any elements it bounds. Thus in the plane, there are three distinct types of minimal neighbourhood: a cell on its own, an edge with the two cells it bounds, and a vertex with the four edges and four cells which it bounds. A set of primitive elements is regarded as open so long as it contains the minimal neighbourhood of each of its members.

5 Closure Spaces

In this section we present a general framework within which the main types of discrete space can be described, and which provides a simple and natural definition of continuity that can be applied to these spaces.

A *closure space* consists of a set U of primitive elements together with a function $cl : 2^U \to 2^U$ satisfying the conditions

1. $cl(\emptyset) = \emptyset$
2. $\forall X \subseteq U : X \subseteq cl(X)$
3. $\forall X, Y \subseteq U : cl(X \cup Y) = cl(X) \cup cl(Y)$.

The function cl associates with each subset $X \subseteq U$ a subset $cl(X) \subseteq U$, called the *closure* of X. Closure spaces were introduced by Čech (1966); they were suggested by Smyth (1995) for the present purpose of unifying various disparate approaches to discrete space. Smyth introduced closure spaces as a restriction of a wider class of *neighbourhood spaces* in which the "closure" function need not satisfy (2). Indeed, Hammer (1955) had already initiated a study of generalised closure functions, and suggested (Hammer 1964) that the most appropriate framework within which to develop a general theory of continuity was provided by a study of what he called *expansive functions* on $\mathcal{P}(\mathcal{U})$, which satisfy (1) and (2) but have in place of (3) the weaker condition $X \subseteq Y \to cl(X) \subseteq cl(Y)$. In this paper we shall restrict our attention to closure functions as defined above, as being most pertinent to the applications we have in mind.

The *interior* operator is dual to the closure:

$$int(X) \triangleq U \setminus cl(U \setminus X).$$

An immediate consequence of this definition is that $int(X) \subseteq X$.

If to the axioms for a closure space is added the rule

$$\forall X \subseteq U : cl(cl(X)) = cl(X),$$

making closure an idempotent operation, then the resulting space is a topological space. Topological spaces include an important class of discrete spaces, called incidence spaces, which we discuss in some detail below.

A subset $X \subseteq U$ is a *neighbourhood* of a point $x \in U$, written $N(X, x)$, so long as $x \in int(X)$. A closure space is *discrete* if every point has a minimal neighbourhood, that is

$$\forall x \in U \exists X \subseteq U \forall Y \subseteq U : N(Y, x) \leftrightarrow X \subseteq Y.$$

In this case we write N_x for the unique minimal neighbourhood of x.

In the next two sections we look at two special subclasses of closure spaces obtained by imposing extra conditions.

6 Adjacency Spaces

6.1 Definition and properties

An adjacency space is a set U endowed with an irreflexive, symmetric relation $A \subseteq U^2$ called *adjacency*. An adjacency space is viewed as a closure space by defining, for $X \subseteq U$,

$$cl(X) \triangleq X \cup \{y \in U \mid \exists x \in X : yAx\}.$$

Then we have also

$$int(X) = \{x \in X \mid \forall y \in U(yAx \rightarrow y \in X)\}$$

An important, easily-proved property of adjacency spaces, not shared by closure spaces in general, is

$$cl(int(X)) \subseteq X \subseteq int(cl(X)).$$

Hence, if $x \in int(X)$ then $cl(\{x\}) \subseteq cl(int(X)) \subseteq X$, so any neighbourhood of x contains $cl(\{x\})$. But also $x \in int(cl(\{x\}))$, so $cl(\{x\})$ is itself a neighbourhood of x. It is therefore the minimal neighbourhood of x. Thus an adjacency space, viewed as a closure space, is necessarily discrete.

In adjacency space we have little use for the notions of open and closed sets. It is easily proved that $cl(X) \subseteq Y \leftrightarrow X \subseteq int(Y)$; if we follow the standard definitions and call a set open if it is equal to its own interior and closed if it is equal to its own closure, then this has the curious consequence that in adjacency space a set is open if and only if it is closed! Such a set is *isolated* since it is not connected to its complement. If the universe U is connected then the only open or closed sets are U itself and the empty set.

An important difference between adjacency spaces and topological spaces is that whereas in the latter the closure operation is idempotent, in the former it is not. Because of this, each region in an adjacency space gives rise to a sequence of closures:

$$X, cl(X), cl(cl(X)), cl(cl(cl(X))), \ldots, cl^n(X), \ldots$$

which may or may not stabilise after a finite number of steps. This gives us a natural measure of the distance of a cell from a region: the distance of cell x from region X is the smallest n for which $x \in cl^n(X)$. This is zero if and only if $x \in cl^0(X) = X$, and is undefined if x and X lie in different connected components of the space. We can now define the *discrete Hausdorff distance* between two regions as the maximum distance of any cell in either region from the other region. For more details, see (Galton 1999).

6.2 Examples of Adjacency Spaces

6.2.1 Rectangular grids

Both the 4-connected rectangular grid, $(\mathbb{Z}^2, A_4)$ and the 8-connected rectangular grid, $(\mathbb{Z}^2, A_8)$ are adjacency spaces. In both these cases we have "connectivity paradoxes". To avoid these, Rosenfeld used a rectangular grid divided into a "foreground" set F and a "background" set $\mathbb{Z}^2 \setminus F$, with adjacency defined by

$$xAy \triangleq \begin{cases} xA_8y & (\text{if } x, y \in F) \\ xA_4y & (\text{otherwise}) \end{cases}$$

(Alternatively, one could reverse the roles of A_4 and A_8 in this definition.) This is also an adjacency space.

In three dimensions, we have the 6-connected cubic grid, defined as $\mathbb{Z}^3$ with (x, y, z) adjacent to the six points $(x \pm 1, y, z), (x, y \pm 1, z), (x, y, z \pm 1)$. Other adjacency relations on the cubic grid (A_{18} and A_{26}) are defined in the obvious way (Kong & Rosenfeld 1989).

6.2.2 Raster regions

An alternative approach to the theory of regions on a rectangular grid is that of (Egenhofer & Sharma 1993). They work with a notion of "region" that is radically different from ours. A region — which they call a *raster region* — is specified by means of a simple closed curve (essentially a Jordan 4-curve). The curve itself is called the boundary of the region, and of the two components of the complement of the curve, the finite one is called the interior of the region, and the other is the exterior. It seems to be immaterial whether the region itself is identified with the interior or with the union of the interior with the boundary; what is important is that it is the boundary that comes first, rather than a set of pixels identified with the region itself. The difference between these raster regions and the regions in our adjacency spaces (which are just arbitrary sets of pixels) can be illustrated thus: if we want to draw a region on a sheet of paper, we can do so either by drawing a closed curve or by shading an area of the paper. The former corresponds to an Egenhofer-Sharma region, the latter to the kinds of region discussed in this paper. These are illustrated in Figure 3.

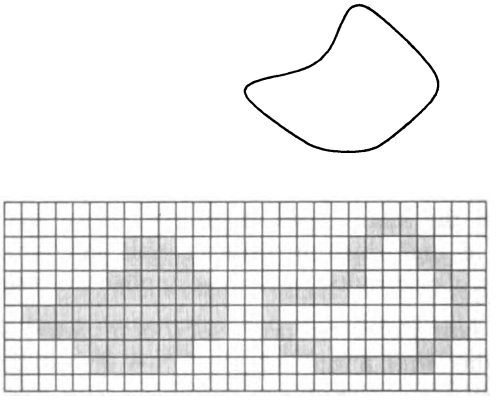


Figure 3: *Two ways of specifying a region in $\mathbb{R}^2$, and their analogues in $\mathbb{Z}^2$. The figure at bottom right corresponds to the raster regions of Egenhofer and Sharma (1993).*

6.2.3 Regions in networks

The theory of adjacency spaces is closely related to some work of Stell & Worboys (1997). They consider a class of algebraic structures called bi-Heyting algebras, which generalise Boolean algebras. The examples they give include regions in networks, where a network is defined as an undirected graph, where loops and multiple edges are allowed. A region in such a network is a subgraph specified by means of a set of nodes and a set of edges, with the proviso that an edge can only be included if the nodes at its endpoints are included as well. Two kinds of "complement" are defined for a region: the *negation* consists of all the nodes not in the region, together with any edges between those nodes, while the *supplement* consists of all the edges not in the region, together with all their end-nodes and any other nodes not in the region. These definitions cannot be transferred in full generality to the setting of adjacency spaces, since two points in an adjacency space are either adjacent or not, whereas two nodes in a network may have any number of distinct edges connecting them. If we restrict this treatment to simple graphs, in which at most one edge can exist between two given nodes, and restrict regions to "regular" subgraphs, in which an edge is present whenever its end-nodes are present, then the set of nodes forms an adjacency space, with two nodes adjacent if and only if they are joined by an edge. The closure and interior operations are related to the bi-Heyting algebra operations quite simply: $cl(X)$ turns out to be supplement of the negation of X, while $int(X)$ is the negation of the supplement.

7 Incidence Spaces

7.1 Definition and properties

An incidence space is a set U equipped with an irreflexive, transitive relation $B \subseteq U^2$, called *bounding*. An incidence space can be viewed as a closure space by defining, for $X \subseteq U$,

$$cl(X) \triangleq X \cup \{y \in U \mid \exists x \in X : yBx\}.$$

The closure space axioms are easily verified.

From the transitivity of B it quickly follows that $cl(cl(X)) = cl(X)$, so closure is idempotent. An incidence space is thus a topological space; its open sets are those $X \subseteq U$ for which $int(X) = X$. Our incidence spaces are essentially identical with the *cellular complexes* of (Kovalevsky 1989). The topology of such spaces has been called the *star-topology* (Ahronovitz et al. 1995).

In an incidence space there is an important connection between bounding and neighbourhoods. Let xBy, and suppose X is not a neighbourhood of y. Then $y \in cl(U \setminus X)$, so yBz for some $z \in U \setminus X$. Since bounding is transitive, we have xBz, so $x \in cl(U \setminus X)$, and hence X is not a neighbourhood of x either. Thus if x bounds y, every neighbourhood of x is also a neighbourhood of y. Now put $N = \{x\} \cup \{y \in U \mid xBy\}$. Then x does not bound any element of $U \setminus N$, so $x \notin cl(U \setminus N)$, so $x \in int(N)$, so N is a neighbourhood of x. Moreover, by the previous result, any neighbourhood of x must be a neighbourhood of each element of N and therefore must include N. Thus N is the minimal neighbourhood N_x of x, and we have proved that an incidence space, considered as a closure space, is necessarily discrete.

Whenever x bounds y the neighbourhoods of x are all neighbourhoods of y. Hence, so long as the bounding relation is not empty, the space fails to satisfy the T_1 separation axiom, which says that for any two points, each has a neighbourhood that is not a neighbourhood of the other. On the other hand, it *does* satisfy the T_0 axiom, which says that no two distinct points have exactly the same neighbourhoods. For if x and y have the same neighbourhoods, they have the same minimal neighbourhood, so $N_x = N_y$, i.e,

$$\{x\} \cup \{z \in U \mid xBz\} = \{y\} \cup \{z \in U \mid yBz\},$$

so if $x \neq y$ then both xBy and yBx. By the asymmetry of B, it follows that $x = y$. Thus an incidence space, viewed as a topological space, is T_0 but not T_1.

In any topological space which is T_0 but not T_1 we can define a strict partial order

$$xBy \triangleq x \neq y \wedge \forall X(Open(X) \wedge x \in X \rightarrow y \in X).$$

(This is the irreflexive part of the *specialisation order*, see (Johnstone 1982).) We can treat this as the bounding relation of an incidence space and consider the topology thereby induced. This will be the same as the original topology if every set of the form $X \cup \{y \in U \mid \exists x \in X(xBy)\}$ is open in that topology.

7.2 Examples of Incidence Spaces

7.2.1 The digital plane

This is equivalent to the *hyper-raster* of (Winter 1995); it is isomorphic to the Khalimsky space on $\mathbf{Z}^2$ (Khalimsky et al. 1990, Smyth 1992). The topology contains four types of element, as follows (throughout, $m, n \in \mathbb{Z}$):

- *Vertices*, notated $V_{mn} = (m, n)$;

- *Vertical edges*, notated $E^v_{mn} = (m, (n, n+1))$;
- *Horizontal edges*, notated $E^h_{mn} = ((m, m+1), n)$;
- *Cells*, notated $C_{mn} = ((m, m+1), (n, n+1))$.

The bounding relation is defined by the rules:

- V_{mn} bounds $E_{(m-1)n}$, $E_{(m+1)n}$, $E_{m(n-1)}$, $E_{m(n+1)}$, $C_{(m-1)(n-1)}$, $C_{m(n-1)}$, $C_{(m-1)n}$ and C_{mn}.
- E^v_{mn} bounds $C_{(m-1)n}$ and C_{mn}.
- E^h_{mn} bounds $C_{m(n-1)}$ and C_{mn}

Thus vertices bound both edges and cells, edges bound cells, and cells do not bound anything. A portion of this space is portrayed in the upper illustration of Figure 4. The lower illustration portrays in schematic form the "real" underlying space, with vertices, edges and cells shown with an indication of their true dimensionality.

In the topology induced by this bounding relation, each individual cell forms a singleton open set; the minimal neighbourhood of an edge consists of the edge together with the two cells it bounds; and the minimal neighbourhood of a vertex consists of the vertex together with the four edges and the four cells which it bounds. An open set cannot contain "naked" edges or vertices: any edge or vertex in the set must be "cushioned" from the exterior of the set by the surrounding cells.

This topology can be obtained from the standard topology on $\mathbb{R}^2$ by means of a simple mapping. For $x \in \mathbb{R}$, define

$$[x] = \begin{cases} x & \text{if } x \in \mathbb{Z} \\ (\lfloor x \rfloor, \lceil x \rceil) & \text{otherwise} \end{cases}$$

Then the mapping we require is $D : (x, y) \to ([x], [y])$. This is homomorphic in the sense that the open sets in the resulting topology are precisely the images of open sets in the topology on $\mathbb{R}^2$. An exactly analogous construction can be performed on $\mathbb{R}^n$ for any $n > 0$.

7.2.2 The Regional Connection Calculus

The Regional Connection Calculus (RCC) is a system for representing and reasoning about qualitative topological relationships between regions in space (Randell et al. 1992). The eight basic relations handled by the calculus are

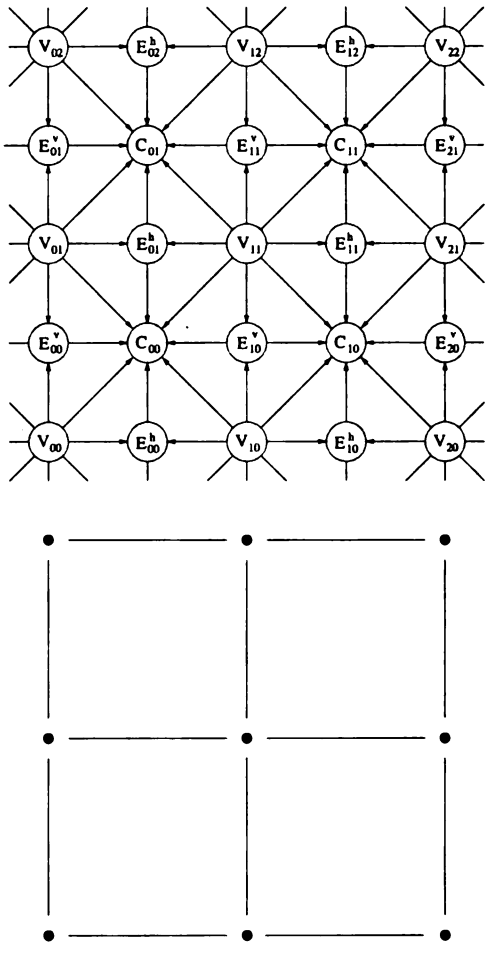


Figure 4: *A portion of a T_0 space defined on a rectangular grid (arrows denote bounding).*

DC	Disconnection
EC	External connection
PO	Partial overlap
EQ	Equality
TPP	Tangential proper part
NTPP	Non-tangential proper part
TPPI	Inverse of TPP
NTPPI	Inverse of NTPP

The conceptual neighbourhood diagram (Figure 5, ignoring the arrowheads) indicates which relations are "adjacent" to each other, in the sense that it is possible by continuous movement to pass from one relation to the other without passing through any intermediate relations.

The discrete set of RCC relations can be viewed as a homomorphic image of the full space of possible region-pairs in $\mathbb{R}^n$. There are in fact many different ways of

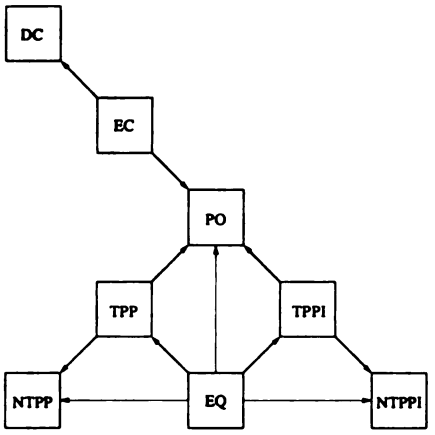


Figure 5: *The Regional Connection Calculus.*

doing this. One way is described in (Galton 1997*b*); for brevity we illustrate here a simpler construction, which applies to the restricted case in which the regions are spheres freely moving about in $\mathbb{R}^3$. The position of a sphere at an instant is uniquely specified by the coordinates (x, y, z) of its centre, together with its radius r; a pair of spheres can thus be represented by a point in an 8-dimensional space. For the purpose of determining the qualitative, RCC relation between two spheres, however, all that matters is r_1 and r_2, the radii of the two spheres, and d, the distance between their centres. In fact, assuming that the spheres always have positive radii, never shrinking to a point, it is enough to consider just the two ratios $x = r_1/r_2$ and $y = d/r_2$. Thus we can first homomorphically collapse the 8-dimensional space down to a two-dimensional space parametrised by x and y. The eight relations can now be expressed in terms of x and y as follows (Figure 6):

DC	$y > x + 1$
EC	$y = x + 1$
PO	$\|1 - x\| < y < x + 1$
EQ	$x = 1, y = 0$
TPP	$0 < y = 1 - x$
NTPP	$y < 1 - x$
TPPI	$0 < y = x - 1$
NTPPI	$y < x - 1$

The topology of Figure 6 carries over to the discrete image to give rise to a bounding relation which is indicated by means of the arrows in Figure 5, thus revealing the space of RCC relations to be an incidence space. Thus, for example, EC bounds PO, since the line of points corresponding to EC forms part of the boundary of the open area corresponding to PO. This fact is of little significance in a static view, but once we consider motion, it provides an important constraint on the possible patterns of temporal incidence of the RCC relations. This issue is discussed more fully below.

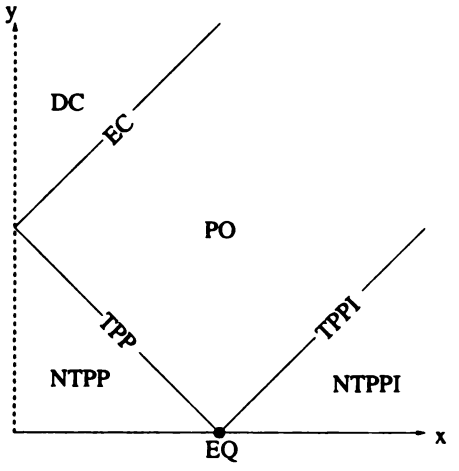


Figure 6: *A two-dimensional image of the space of possible relations between two spheres.*

Of course, the RCC diagram applies to more than just spheres. If we allow arbitrary regions, then the number of degrees of freedom in the full quantitative space becomes infinite; yet still we can, I take it, affirm that the RCC diagram is a homomorphic image of this infinite-dimensional space, faithfully reflecting its topology in the T_0 bounding relation.

8 Continuous Motion in Discrete Space

The natural way to generalise the topological definition of continuity to the more general setting of closure spaces is as follows:

> A function f from closure space U_1 to closure space U_2 is continuous so long as
>
> $$\forall X \subseteq U_1 : f(cl(X)) \subseteq cl(f(X)).$$

This will provide us with the means to specify continuous motion in a space which is modelled as a closure space, so long as time is also so modelled.

8.1 Continuous motion of an atomic object

We begin with the simplest case, the continuous motion of an atomic object. By an atomic object is understood an object whose position at any given time belongs to the set of primitive spatial elements of the

space in which it exists. Thus the motion of an atomic object a in a space S over a time T is specified by a function $pos_a : T \to S$. We assume that both T and S are closure spaces; we shall investigate three different types of closure space here, namely adjacency spaces, incidence spaces, and continuous spaces. Since each of S and T can be any of these three, there are nine different space-time combinations to examine, which we label AA, AI, AC, IA, II, IC, CA, CI, and CC, where the first letter specifies the type of closure space used to model time, and second the type used for space. As we shall see, not all of the nine varieties of space-time give rise to interesting forms of continuous motion.

For simplicity, we assume that space is one-dimensional. This enables us to portray the motion of an atom as a world-line in a space-time diagram. Representative diagrams for each type of space-time are shown in Figure 7; time is plotted along the vertical axis, space along the horizontal, in accordance with the usual practice for such diagrams (but contrary to the other widespread practice of plotting the independent variable along the horizontal axis).

We shall use the following standard one-dimensional models of each of the three types of space:

1. An adjacency space is modelled by $\mathbb{Z}$, with xAy defined as $|x - y| = 1$.
2. An incidence space is modelled by $\mathbb{Z} \cup \{(n, n+1) \mid n \in \mathbb{Z}\}$, with xBy defined as $x \in \mathbb{Z}, y \in \{(x-1, x), (x, x+1)\}$. Elements of type n will be called nodes (in space) or instants (in time), while those of type $(n, n+1)$ will be called edges (in space) or intervals (in time).
3. A continuous space is modelled by $\mathbb{R}$.

We use the same model for time and the one-dimensional space (but a number x, considered as a time point, is to be regarded as distinct from the same number considered as a space point). We briefly explain the main features of each of the nine types:

AA. For $t \in \mathbb{Z}$, we have $cl(t) = \{t-1, t, t+1\}$, so the continuity condition requires that $pos_a(t \pm 1) \in \{pos_a(t) - 1, pos_a(t), pos_a(t) + 1\}$. Thus a continuously moving atom cannot move more than one space step in one time step. This is the "quasi-continuity" of (Galton 1997*b*). An AA space-time has an inbuilt maximum speed (the "speed of light").

AI. If $pos_a(t_0) = n$, then by continuity, $pos_a(t_0 \pm 1) \in cl(\{n\}) = \{n\}$, and hence $pos_a(t) = n$ for all t. For an object at a node, continuous motion means staying put! Likewise, if $pos_a(t_0) = (n, n+1)$, then $pos_a(t_0 \pm 1) \in \{n, (n, n+1), n+1\}$. We already know that n and $n+1$ are ruled out, hence $pos_a(t_0 \pm 1) = pos_a(t_0)$. Thus all continuous motion in AI space-time is staying put. All non-trivial motion is discontinuous. (The illustration for this case shows two world-lines, one for an object stuck at a point, the other for an object stuck on an edge.)

AC. As in AI, all motion is discontinuous.

IA. If $pos_a((t, t+1)) = n$, then $pos_a(t)$ and $pos_a(t+1)$ must be in $\{n-1, n, n+1\}$; $pos_a(t)$ imposes no further restrictions. The motion does not look very much different from the AA case: we can convert time into an adjacency space by putting xAy whenever any of $x = y$, xBy, or yBx holds, and the possibilities for the diagram remain essentially unchanged. The difference between the two cases is one of interpretation: in IA, we can have our object moving from one position to a neighbouring position for an single instant before moving away. This is odd because it is natural to interpret the instants in incidence-time as durationless, as against adjacency-time where the temporal atoms are most naturally regarded as having some minimal, but positive, duration.

II. Here things start to become more interesting. There are four types of spatio-temporal occupancy to consider: the position at an instant may be either a node or an edge, and the position at an interval may be either a node or an edge. By continuity, if $pos_a((t, t+1)) = n$ then $pos_a(t) = pos_a(t+1) = n$ also. It follows that if $pos_a(t) = (n, n+1)$ then $pos_a((t-1, t)) = pos_a((t, t+1)) = (n, n+1)$. It will be observed that both these constraints are satisfied by the world-line illustrated for this case.

IC. As in AI and AC, all motion is discontinuous.

CA. This case is rather similar to IA, with the difference that the intervals can be of arbitrary size.

CI. This is similar to II, but again with intervals of arbitrary size.

CC. The familiar case of continuous motion along a line being modelled by means of a continuous function from $\mathbb{R}$ to $\mathbb{R}$.

In summary, the cases AI, AC, and IC are of no practical interest; cases IA and CA are strange in that they allow instantaneous visits to cells of the adjacency space. CC is of fundamental importance to science, but belongs outside the sphere of interest of the present paper because the space is continuous. Thus the only cases that will be of further concern to us are AA, II, and CI.

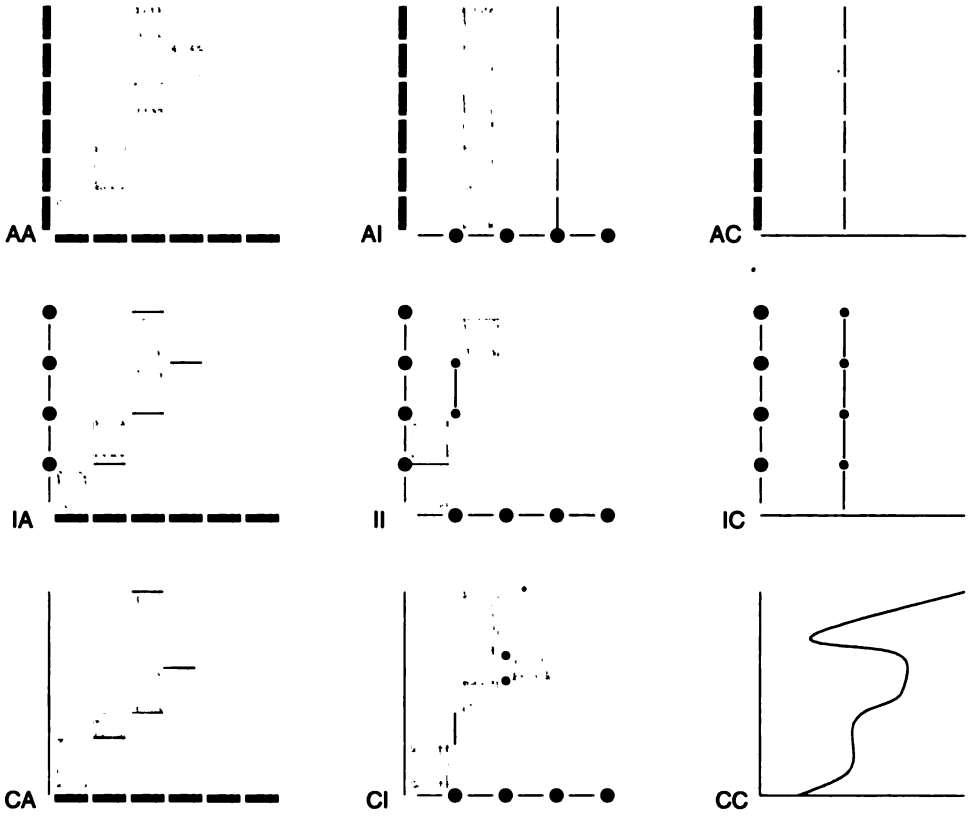


Figure 7: *Continuous motion of an atom in one-dimensional space.*

Although the discussion has focussed on one-dimensional space, the conclusions apply to continuous motion in spaces of more than one dimension too. The reader may derive some edification from attempting to draw some possible world-lines in two-dimensional spaces for the cases we have singled out as of interest.

Of the discrete spaces we have discussed, there is one type in particular for which it is natural to consider the motion of an atom. This is the space of qualitative relations such as RCC, as discussed in §7.2.2. The "space" in question is a conceptual space, dependent on the physical space within which the regions are located, but completely distinct from it. It is natural to consider motion in this space over continuous time, thus the scheme CI best describes the set-up here.

In §7.2.2 we derived the RCC space as a homomorphic image of the space of sphere-pairs in $\mathbb{R}^3$. We will remain with this construction, but our observations will carry over, *mutatis mutandis*, to similar constructions in which RCC space is derived from some more comprehensive continuous space of region pairs.

If we imagine our spheres moving about and changing size continuously, the point representing them weaves a continuous path in the (x, y)-plane of Figure 6. At the same time, the RCC relation between the spheres wanders about the conceptual neighbourhood diagram of Figure 5; and throughout, the topological continuity condition applies: the set of instants on which the position (whether expressed quantitatively or by means of an RCC relation) is contained in some given open set is itself open. Thus, under continuous motion, the set of instants at which the disjunctive relation "DC or EC or PO" holds must be open, since the set {DC,EC,PO} is open in this topology.

Constraints on the possible motions can be read off directly from the arrows in the incidence-space diagram. The interpretation of the arrows is as follows: the presence of an arrow from relation R_1 to relation R_2 means that it is possible for R_1 to hold at a instant which bounds an open interval over which R_2 holds. The fact that the arrow points from EC to DC, and not the other way round, allows one to deduce that it is possible to move continuously from DC to EC and then back to DC in such a way that the EC relation holds only for an instant; but in any continuous motion from EC to DC and then back to EC, the DC relation has to hold over an interval. The diagram

also enables one to derive such comparatively recondite facts as that whereas it is possible for the relation TPP to hold instantaneously in the context of a continuous transition between PO and NTPP, this is not possible in the context of a transition between PO and EQ. In (Galton 1997*b*) conceptual neighbourhood diagrams are augmented with arrows indicating what was there called the "dominance" relation between qualitative states; this turns out to be identical with the bounding relation discussed in the present context.

In this example, "motion" refers to the motion of a notional "point" in a high-level conceptual "space"; this motion reflects underlying changes in the continuous domain which correspond more immediately to happenings in the physical space of our experience. The moving object in the high-level space is always a point; if one were to consider "regions" made up from more than one point, the only sensible way of interpreting this would be that one is modelling the motion of a *set* of region-pairs in the underlying space, or else a set of *possible* motions of a single region-pair.

8.2 Continuous motion of an extended object

We begin by considering the case AA, in which both time and space are represented as adjacency spaces. The position of an extended object in adjacency-space is a set of cells. The motion of such an object is given by a function $pos : \mathbb{Z} \to \mathcal{P}(S)$ specifying its position at each time. For $t \in \mathbb{Z}$ we have $cl(t) = \{t-1, t, t+1\}$, so the condition for *pos* to be continuous is that

$$\forall t \in \mathbb{Z} : pos(\{t-1, t, t+1\}) \subseteq cl(pos(t)).$$

We thus have $pos(t) \subseteq cl(pos(t+1))$ and $pos(t+1) \subseteq cl(pos(t))$. Thus the object's positions at two neighbouring times must each be contained in the other's closure. This is the "almost equal" relation of (Galton 1999), equivalent to a discrete Hausdorff distance not greater than 1. Motion of a region in AA space-time is continuous so long as neighbouring positions differ only with respect to cells on their boundaries.

For the case of II, we consider the digital image of a small open disc in $\mathbb{R}^2$, using the hyper-raster incidence space. The grain size of the image is coarse relative to the circle: this is in order to highlight the bizarreness of some of the phenomena of digital imaging. Figure 8 shows a sequence of positions of the circle such that the corresponding sequence of digital images is complete, i.e., the image passes from each member of the sequence to the next without a break (if the circle were closed, more images would have to be interpolated into the sequence, but these would only be realised on instants). If the space of possible positions of the disc is itself topologised as an incidence space, then the bounding relations on the set of digital images are as shown by the arrows in the figure. Then a possible sequence in accordance with continuity is

Time	(1,2)	2	(2,3)	3	(3,4)	4	(4,5)	5
Image	1	2	3	4	5	6	7	8

The case CI is essentially similar to II, differing only in that it is the motion itself which imposes the segmentation of time into a discrete alternating series of instants and intervals.

9 Conclusions

The work presented here brings together a number of different strands of investigation to provide a synthesis which is, we believe, essentially new. Following a suggestion of Smyth (1995) we adopted the notion of a Closure Space as providing an appropriate framework for investigating various kinds of discrete space. We examined two special cases of the more general notion, adjacency and incidence spaces, and discussed a number of examples of each. We showed how these general systems subsume a number of existing systems, notably the hyper-raster of Winter (itself deriving from work of Kovalevsky), qualitative systems such as RCC based on the idea of conceptual neighbourhood, and the adjacency-based representation of discrete space of (Galton 1999). The Digital Topology of Rosenfeld, the raster-regions of Egenhofer and Sharma, and the network-regions of Stell and Worboys, can also be usefully approached within the framework described here.

We used the natural generalisation of topological continuity to provide a criterion for continuity within the more general framework, and systematically examined the possible types of continuous motion obtained by varying the type of model chosen for representing space and time. We concluded with a discussion of how some of these types of motion appear within particular examples of the general framework.

References

Ahronovitz, E., Aubert, J.-P. & Fiorio, C. (1995), The star-topology: A topology for image analysis, *in* 'Discrete Geometry for Computer Imagery', Groupe GDR PRC/AMI du CNRS, pp. 107–116.

Čech, E. (1966), *Topological Spaces*, John Wiley & Sons. Revised edition by Zdeněk Frolík and Miroslav Katětov.

Egenhofer, M. J. & Sharma, J. (1993), Topological relations between regions in R^2 and Z^2, *in* D. Abel

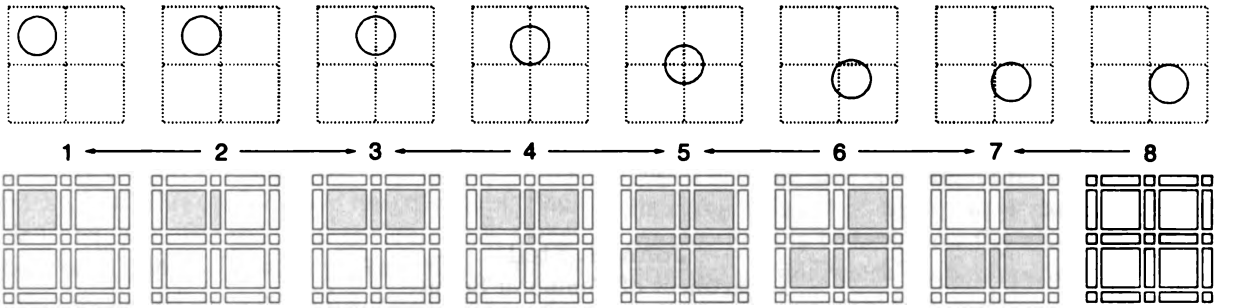


Figure 8: *Positions of a circle in* $\mathbb{R}^2$*, and their digital images.*

& B. C. Ooi, eds, 'Advances in Spatial Databases — Third International Symposium on Large Spatial Databases SSD'93', Vol. 692 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 316–336.

Freksa, C. (1992), 'Temporal reasoning based on semi-intervals', *Artificial Inteliigence* **54**, 199–227.

Galton, A. P. (1997*a*), Continuous change in spatial regions, *in* S. C. Hirtle & A. U. Frank, eds, 'Spatial Information Theory: A Theoretical Basis for GIS', Vol. 1329 of *Lecture Notes in Computer Science*, Springer, pp. 1–13. Proceedings of International Conference COSIT'97.

Galton, A. P. (1997*b*), Space, time and movement, *in* O. Stock, ed., 'Spatial and Temporal Reasoning', Kluwer Academic Publishers, Dordrecht, pp. 321–352.

Galton, A. P. (1999), The mereotopology of discrete space, *in* C. Freksa & D. M. Mark, eds, 'Spatial Information Theory: Cognitive and Computational Foundations of Geographic Science', Springer-Verlag, pp. 251–266. Proceedings of International Conference COSIT'99.

Hammer, P. C. (1955), 'General topology, symmetry, and convexity', *Transactions of the Wisconsin Academy of Sciences, Arts and Letters* **44**, 221–255.

Hammer, P. C. (1964), 'Extended topology: Continuity I', *Portugaliae Mathematica* **25**, 77–93.

Johnstone, P. T. (1982), *Stone Spaces*, number 3 *in* 'Cambridge Studies in Advanced Mathematics', Cambridge University Press, Cambridge, UK.

Khalimsky, E., Kopperman, R. & Meyer, P. R. (1990), 'Computer graphics and connected topologies on finite ordered sets', *Topology and its Applications* **36**, 1–17.

Kong, T. Y. & Rosenfeld, A. (1989), 'Digital topology: Introduction and survey', *Computer Vision, Graphics and Image Processing* **48**, 357–393.

Kong, T. Y., Kopperman, R. & Meyer, P. R. (1991), 'A topological approach to digital topology', *The American Mathematical Monthly* **98**(10), 901–917.

Kovalevsky, V. A. (1989), 'Finite topology as applied to image analysis', *Computer Vision, Graphics, and Image Processing* **46**, 141–161.

Randell, D. A., Cui, Z. & Cohn, A. G. (1992), A spatial logic based on regions and connection, *in* 'Proceedings of the Third International Conference on Knowledge Representation and Reasoning', Morgan Kaufmann, San Mateo, California, pp. 165–176. Cambridge, Massachusetts, October 1992.

Smyth, M. B. (1992), Topology, *in* S. Abramsky, D. M. Gabbay & T. S. E. Maibaum, eds, 'Handbook of Logic in Computer Science', Clarendon Press, Oxford, pp. 641–761.

Smyth, M. B. (1995), 'Semi-metrics, closure spaces and digital topology', *Theoretical Computer Science* **151**, 257–276.

Stell, J. G. & Worboys, M. F. (1997), The algebraic structure of sets of regions, *in* S. C. Hirtle & A. U. Frank, eds, 'Spatial Information Theory: A Theoretical Basis for GIS', Vol. 1329 of *Lecture Notes in Computer Science*, Springer, pp. 163–174. Proceedings of International Conference COSIT'97.

Winter, S. (1995), Topological relations between discrete regions, *in* M. J. Egenhofer & J. R. Herring, eds, 'Advances in Spatial Databases', Springer, Berlin, pp. 310–327.

The Representation of Discrete Multi-resolution Spatial Knowledge

John G. Stell
Department of Computer Science
Keele University
Keele, Staffs, ST5 5BG, U. K.
email: john@cs.keele.ac.uk

Abstract

The representation of spatial knowledge has been widely studied leading to formal systems such as the Region-Connection Calculus, and various formal accounts of mereotopology. The majority of this work has been in the context of spaces which are continuous and infinitely divisible. However, discrete spaces are of practical importance, and arise in several application areas. The present paper contributes to the representation of discrete spatial knowledge on two fronts. Firstly it provides an algebraic calculus for discrete regions, and demonstrates its expressiveness by showing how the concepts of interior, closure, boundary, exterior, and connection can all be formulated in the calculus. The second advance presented is an account of discrete spaces at multiple levels of detail. This is achieved by developing a general theory of coarsening for sets which is then extended to the case of discrete regions.

1 INTRODUCTION

1.1 APPLICATIONS OF DISCRETE SPACES

Discrete spaces are of practical importance in several application areas. Examples include digital image processing, where a simple monochrome image may be described in terms of which pixels are on and which off, and where the spatial structure is determined by which pixels are adjacent to each other. Another example is the manipulation of various kinds of network in geographic information systems (GIS). Geographic data often includes information about road and rail networks, and distribution networks for electricity companies and other utilities. Further examples which motivate the study of discrete spaces can be found within the Internet, both in the physical networks of computers, and in networks of links between web pages. In all of these examples we have discrete spaces which can be modelled as a graph consisting of nodes and edges. In the case of geo-referenced networks, the abstract graph may not model adequately all the features of the data, but it will nevertheless be an important component of the information. Another use of discrete space within GIS arises from partitions of the plane. Any such partition has a dual graph, where the nodes are the cells of the partition and there is an edge between two cells if they share a one-dimensional boundary segment.

In working with applications where discrete spaces arise, it is frequently important to be able to deal with data at a variety of levels of detail. This is especially true in the area of GIS where the 'generalization' of data has been widely studied (Müller, Lagrange & Weibel 1995) and has significant practical applications. In this context, generalization means the process of reducing the level of detail in data, as in the production of small scale maps from large scale ones. Even if we model a geo-referenced network as an abstract graph, the problem of presenting the data at multiple levels of detail is both of practical significance, and not fully understood. Maps of bus routes and underground railway systems in major cities are a case in point here. Although these are conventionally presented at a fixed level of detail, it is easy to imagine applications running on palmtop devices where dynamic control of level of detail in such maps would be an important feature.

The aim of this paper is to propose a formal framework for the representation and manipulation of discrete spatial regions at multiple levels of detail. The areas of application of this formal model are intended to include the various examples mentioned above.

1.2 FORMAL MODELS OF SPACE

Formal systems for the representation of spatial knowledge have been developed from a number of viewpoints. Within qualitative spatial reasoning, one of the most widely studied systems is the Region-Connection Calculus (RCC) due to Cohn and his colleagues (Randall & Cohn 1989, Randall, Cui & Cohn 1992, Cohn, Bennett et al. 1997). From a philosophical perspective various formal accounts of mereotopology have been advanced (Smith 1996, Smith 1997, Asher & Vieu 1995, Borgo, Guarino & Masolo 1996). A detailed survey of mereotopology which also deals with the RCC and related systems can be found in the recent book by Casati & Varzi (1999). Schemes for classifying relationships between spatial regions have arisen from the needs of geographic information systems (GIS) (Egenhofer & Franzosa 1991, Egenhofer & Franzosa 1995, Galton 1998). The majority of all this work has been concerned with space which is infinitely divisible and continuous. For example, the RCC axioms stipulate (Cohn et al. 1997, p283) that every region has a non-tangential proper part, thus ensuring that space is infinitely divisible. The formulation of a version of the RCC axioms having discrete spaces as their models is a problem which has not yet received a satisfactory solution.

Discrete spaces are of practical significance, as the earlier examples have demonstrated, and some papers have addressed the topic of discrete space (Kaufman 1991, Galton 1999, Masolo & Vieu 1999). Another issue which has received relatively little attention is that of spatial representation at varying levels of detail. Although granularity has been studied in AI and logic, for example in van Lambalgen's logic of vision (van Lambalgen & van der Does 1997), and in the theory of rough sets (Orłowska 1998), the topic is under-represented in the literature on space. Some papers which do tackle the topic are (Euzenat 1994, Euzenat 1995, Bittner & Stell 1998), and literature in the GIS context such as (Timpf & Frank 1997, Worboys 1998, Stell & Worboys 1998, Stell & Worboys 1999).

1.3 STRUCTURE OF THE PAPER

The present paper proposes a formal framework for the representation of discrete spatial regions at multiple levels of detail. In presenting this framework, I first describe, in section 2, an algebraic calculus for discrete spatial regions at a fixed level of detail, and show how this relates to Galton's work (Galton 1999). A general theory of sets at different levels of detail is then presented in section 3. This provides a theory of coarse sets which has some connections both to rough sets and to fuzzy sets, but differs from each in certain features. A simple way of extending this material from sets to graphs appears in section 4. In order to develop a more sophisticated way of dealing with graphs at multiple levels of detail, section 5 discusses how the idea of representing subsets of a set by functions to a set of truth values can be extended to graphs. Section 6 shows how the material in section 5 provides a guide which enables operations on detailed graphs to be extended to operations on coarse ones. It is not possible in a short paper to present all the details of the theory, but sufficient detail is given to indicate the main features of the approach. Finally, section 7 presents conclusions and suggests some directions for further work.

2 THE ALGEBRA OF FINE DISCRETE REGIONS

2.1 INTRODUCTION

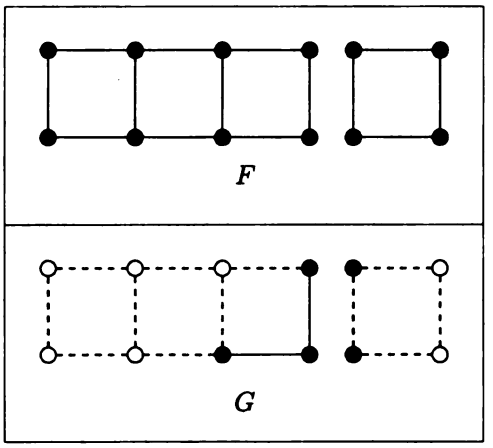
In this section we first review the qualitative approach to discrete space proposed by Galton (1999). This approach takes a discrete space to be a set of cells together with an adjacency relation. Regions in a discrete space are specified by sets of cells. Important spatial concepts, including the closure, interior and boundary of a region, as well as a relation of connection between regions are then defined using standard set-theoretical tools on sets of cells. Having reviewed Galton's approach we present an alternative account based on an algebraic calculus for discrete regions. Technically, the algebraic calculus arises from the bi-Heyting algebra structure on the set of subgraphs of a graph (Reyes & Zolfaghari 1996, Lawvere 1986). Although the possibility of applying this algebraic structure to spatial representation has been suggested already (Stell & Worboys 1997, Stell 1999*b*), the details of its application to discrete spatial regions do not seem to have appeared before.

One advantage of this algebraic calculus is that it promises to allow the extensive work on qualitative treatments of continuous space to be related to the developing theory for discrete space. This is because the point-free approach to qualitative continuous space suggested in (Stell & Worboys 1997) and studied in detail in (Stell 1999*a*) is based on closely related algebraic structures. There is evidence to suggest that an atomic version of the RCC system can be obtained using these structures.

Definition 1 *A* ***discrete space*** (F, adj_F) *consists of a set* F *and a reflexive symmetric relation,* adj_F*, on*

the set F.

Often a discrete space (F, adj_F) will be denoted just by F. The elements of the set are to be thought of as cells and the relation represents adjacency. Diagrams of discrete spaces may be given as follows.



In the upper diagram above we have a discrete space F consisting of 12 cells. The lines in the diagram represent the adjacency relation adj_F. Note that as the relation is assumed to be reflexive there are effectively loops on each node, but as there are no nodes without loops it is not necessary to draw these explicitly.

The lower diagram above shows a part of the space F, which is denoted G. The cells and adjacencies included in G are indicated by the solid discs and lines in the diagram. Parts of this form, which are called subspaces, may have nodes which are adjacent in F but not adjacent in G.

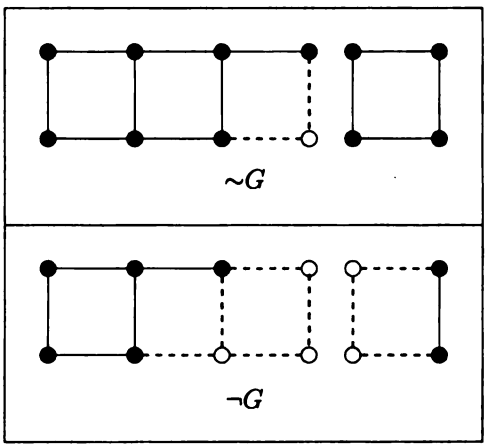
Definition 2 *A **subspace** of a discrete space (F, adj_F) is a discrete space (G, adj_G) where $G \subseteq F$, and for all $g_1, g_2 \in G$, $\mathsf{adj}_G(g_1, g_2)$ implies $\mathsf{adj}_F(g_1, g_2)$. Subspaces satisfying the additional condition $\mathsf{adj}_F(g_1, g_2)$ implies $\mathsf{adj}_G(g_1, g_2)$ are said to be **regions**.*

Note that a region, but not an arbitrary subspace, can be specified just by giving a set of cells of F. The subspaces of a discrete space include the empty subspace, denoted $\bot$, and the whole space, denoted $\top$.

2.2 OPERATIONS ON SUBSPACES

If X is a set, the set of all subsets of X forms a Boolean algebra with the operations of union, intersection, complement, and the distinguished elements X and $\varnothing$. In the case that X is a discrete space, there are analogous operations to union and intersection but we have two distinct types of complement. The union of two subspaces is easy to visualize in terms of the diagrams used above: the union of G and H is depicted by a diagram having the union of the sets of nodes and the union of the sets of edges from the diagrams for G and H. Similarly, the diagram for the intersection is obtained by intersecting the sets of nodes and of edges.

The two kinds of complement arise from the existence of two kinds of elements in a subspace: nodes and edges. The **negation** of G, $\neg G$, is obtained by taking the complement of the cells in G, and setting two such cells to be adjacent in $\neg G$ if they are adjacent in F. The **supplement** of G, $\sim G$, is obtained by taking the complement of the edges in G, and taking the cells to be all the cells in F which are endpoints of these edges. The following diagram shows the negation and the supplement when G is the subspace which appeared earlier.

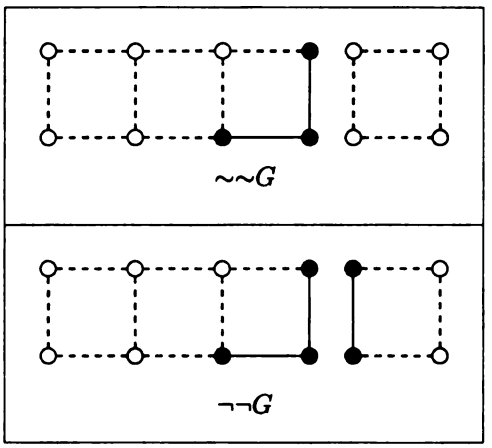


Some important properties of $\neg$ and $\sim$ are summarized in the following lemma.

Lemma 1 *For any subspace G*

$$G \vee {\sim}G = \top \qquad {\sim}{\sim}{\sim}G = {\sim}G$$
$$G \wedge \neg G = \bot \qquad \neg\neg\neg G = \neg G$$

However, it is important to be aware that many properties of the complement operation on subsets do not hold for these operations on subspaces. In particular, the equations $\neg\neg G = G$ and ${\sim}{\sim}G = G$ are false in general, as the diagrams below demonstrate.



Other equations that are false in general include both $G \vee \neg G = \top$ and $G \wedge \sim G = \bot$.

Lemma 2 *A subspace G is a region if and only if* $\neg\neg G = G$.

This is exactly analogous to the situation in continuous spatial reasoning where in a topological space we often do not deal with arbitrary closed sets but restrict attention to the regular closed ones. In the discrete case the operation $G \mapsto \neg\neg G$ can be called the **regularization** of G.

2.3 EXPRESSIVENESS OF THE OPERATIONS

To demonstrate the expressiveness of the operations introduced above, we will show how they can be used to formulate several important concepts. First we recall some definitions from (Galton 1999). These definitions make use of the notion of the **neighbourhood** of a cell, x, in a discrete space F. The neighbourhood of x is denoted N_x and is defined to be $\{y \in F \mid \mathrm{adj}_F(x, y)\}$. The following concepts are defined for regions G and H:

Interior $G^\circ = \{x \in F \mid N_x \subseteq G\}$

Exterior The exterior of G is $(F - G)^\circ$.

Closure $\overline{G} = \{x \in F \mid N_x \cap G \neq \varnothing\}$

Boundary $\partial G = \{x \in F \mid N_x \cap G \neq \varnothing$ and $N_x - G \neq \varnothing\}$

Connection $\mathsf{C}(G, H) = \exists x, y \in F$ $x \in G$ and $y \in H$ and $\mathrm{adj}_F(x, y)$.

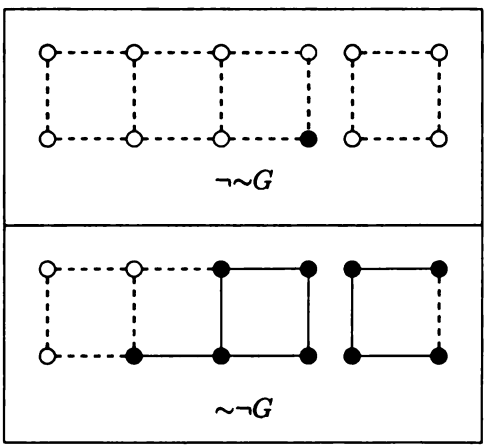
These terms do have some very close analogies with the same terms in conventional topology. However, there are also some important differences in the properties they share. Since the intention here is to re-formulate Galton's approach, his terminology has been adopted.

The above concepts can all be described in a straightforward way in our algebraic calculus.

Theorem 3 *When interior, exterior, closure, boundary and connection are defined as above,*

1. *The interior of G is $\neg\sim G$.*
2. *The exterior of G is $\neg\sim\neg G$.*
3. *The closure of G is $\sim\neg G$.*
4. *The boundary of G is $G \wedge \sim G$.*
5. *G and H are connected if and only if $\sim\neg G \wedge \sim\neg H \neq \bot$.*

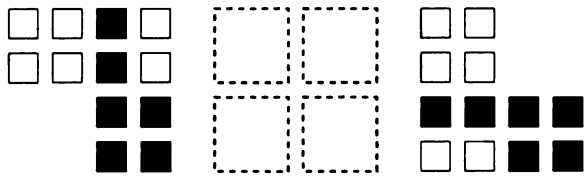
The interior and closure are illustrated below.



3 COARSENING SETS

3.1 EXAMPLE

Suppose there are two sources of information about a spatial region. These might be two human observers or the information might be derived by remote sensing, e.g. sattelite images of the earth. Each observer provides information about a region by describing which cells are in the region and which are not. The information from the observers will be provided at a certain level of detail, but it may be necessary to combine the two observations and present the resulting data at a coarser level of detail. This is illustrated in the following diagram.



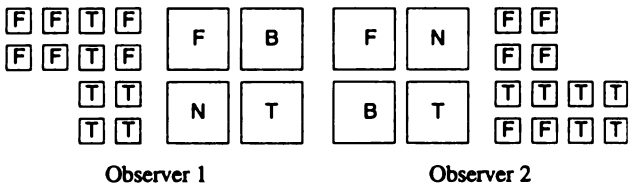
Observer 1 Coarser level of detail Observer 2

The framework with respect to which the first observer provides information consists of 12 cells. Denote this set by X. Note that at this stage it is assumed that X is just a set, the introduction of an adjacency relation on the cells will be treated later in section 6. The coarser level of detail consists of a framework of 4 cells, and this set will be denoted by Y. There is a function $f : X \to Y$ taking each cell in X to the larger cell in Y of which it is a part. It is a significant and novel feature of this work that f is not assumed to be a surjective (or onto) function. That is there may be cells at the coarser level of detail which do not have corresponding cells at the lower level. If f were required to be surjective, then the coarser level of detail would be obtained by imposing an equivalence relation on the finer level of detail. This restriction would lead to the theory of rough sets (Orłowska 1998), and related work (Worboys 1998).

It is now necessary to consider how the observation on X can be represented at the coarser level of detail. The observation itself is a subset of X. A subset of X is a classification of the elements of the set into two classes: *"in the subset"* and *"not in the subset"*. Thus using T and F to stand for true and false respectively, subsets of X correspond to functions from X to $\{\mathsf{T},\mathsf{F}\}$. The representation of the observation $\alpha : X \to \{\mathsf{T},\mathsf{F}\}$ at the lower level of detail is given by a function from Y to a suitable set of classifying values. These classifying values which will here be taken to be T (true), F (false), B (both), and N (neither), correspond to the truth values of the four valued logic introduced by Belnap (1977), and studied further by Arieli & Avron (1998). The coarsening of the observation $\alpha : X \to \{\mathsf{T},\mathsf{F}\}$ is a function $\alpha \boxplus f : Y \to \{\mathsf{T},\mathsf{F},\mathsf{B},\mathsf{N}\}$ defined as follows.

$$(\alpha \boxplus f)y = \begin{cases} \mathsf{T} & \text{if } \{\alpha x \mid fx = y\} = \{\mathsf{T}\} \\ \mathsf{F} & \text{if } \{\alpha x \mid fx = y\} = \{\mathsf{F}\} \\ \mathsf{B} & \text{if } \{\alpha x \mid fx = y\} = \{\mathsf{T},\mathsf{F}\} \\ \mathsf{N} & \text{if } \{\alpha x \mid fx = y\} = \{\,\} \end{cases}$$

The effect of this on each observation is illustrated below.



Observer 1 Observer 2

The use of four truth values is clearly needed here as we need to distinguish two values intermediate in truth between T and F. If a given cell at the coarser level is labelled B for one observation and T for another we have some conflict between the observations. But a cell labelled N in one and T in another observation does not result in any conflict.

3.2 GENERAL THEORY

In this subsection a general theory of coarsening observations will be presented. This does not assume that the detailed observations are classified by functions to $\{\mathsf{T},\mathsf{F}\}$, we will work in a more general context which allows multi-valued logics.

3.2.1 Main Features

Some key features of the theory are illustrated in the following diagram.

$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\alpha \boxplus f} \\ \Omega & \xrightarrow{\eta} & \mathcal{A}\Omega \end{array}$$

In the diagram Ω is a complete lattice of truth values equipped with a negation, $-$, satisfying $--\omega = \omega$. The greatest and least elements of Ω are denoted by $\top$ and $\bot$ respectively, and the ordering in Ω will be written $\leq$. The diagram concerns two levels of detail. At the finer level there is a set X and α gives an Ω-valued subset of X. The set Y is a coarser description of X, and the relationship between X and Y is given by the function f. This function induces an equivalence relation, $\approx$, on X where $x_1 \approx x_2$ iff $fx_1 = fx_2$. The interpretation of $\approx$ is that elements of X are equivalent iff they are indistinguishable at the coarser level of detail. There is no assumption that f is surjective, so f and Y are not determined by $\approx$, as would be the case in the theory of rough sets.

The structure $\mathcal{A}\Omega$ is a bilattice, in the sense of (Ginsberg 1988), the elements of which represent coarsened versions of the truth values in Ω. Details of the theory of bilattices can be found in (Ginsberg 1988, Arieli & Avron 1996), but the main features we need here are that $\mathcal{A}\Omega$ carries two orderings, and a negation which reverses one of the orderings while preserving

the other. The two orderings will be denoted $\leq_t$ and $\leq_c$, these compare respectively degrees of truth and degrees of coarseness. The coarse truth values in $\mathcal{A}\Omega$ include the finer ones in Ω. This embedding of Ω in $\mathcal{A}\Omega$ is given by the function $\eta : \Omega \to \mathcal{A}\Omega$.

Each Ω-valued subset, α, at the finer level of detail has a best possible (i.e. least coarse) representation at the coarser level. This coarser representation is an $\mathcal{A}\Omega$-valued subset of Y denoted $\alpha \boxplus f$, which we read as 'α coarsened by f'.

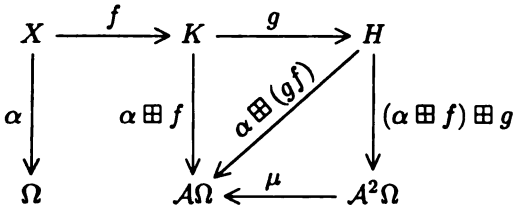
3.2.2 Technical Details

The elements of $\mathcal{A}\Omega$ are intervals in Ω. Thus every element of $\mathcal{A}\Omega$ is a subset of Ω of the form $[a, b]$ where $[a, b] = \{\omega \in \Omega \mid a \leq \omega \leq b\}$. The empty interval, which occurs when $b > a$, is permitted. We will assume that when $[a, b]$ is written either $a \leq b$, or that $a = \top$ and $b = \bot$. The embedding $\eta : \Omega \to \mathcal{A}\Omega$ is the mapping $\omega \mapsto [\omega, \omega]$. The two orderings $\leq_t$ (truth) and $\leq_c$ (coarseness) are defined by

$$\begin{array}{lll} [a,b] \leq_t [c,d] & \text{iff} & a \leq c \text{ and } b \leq d, \\ [a,b] \leq_c [c,d] & \text{iff} & [a,b] \subseteq [c,d], \end{array}$$

and the negation is given by $-[a, b] = [-b, -a]$. The coarsening of α is defined by $\alpha \boxplus f = \bigoplus\{g \in (\mathcal{A}\Omega)^Y \mid \eta\alpha \leq_c gf\}$, where $\oplus$ denotes the join in the lattice $(\mathcal{A}\Omega, \leq_c)$, $(\mathcal{A}\Omega)^Y$ denotes the set of functions from Y to $\mathcal{A}\Omega$, and the ordering $\leq_c$ on $\mathcal{A}\Omega$ has been extended to $(\mathcal{A}\Omega)^Y$ by $g_1 \leq_c g_2$ iff $g_1 y \leq_c g_2 y$ for all $y \in Y$.

A coarse set taking values in $\mathcal{A}\Omega$ can be coarsened again to one taking values in $\mathcal{A}^2\Omega = \mathcal{A}\mathcal{A}\Omega$. There is a function, μ, taking elements of $\mathcal{A}^2\Omega$ to $\mathcal{A}\Omega$ defined by $\mu[[a, b], [c, d]] = [a, d]$. When a subset α is subjected to two sucessive coarsenings we have the following situation



There are two coarse versions of α at the level of detail of H which are related as follows.

Lemma 4 $\alpha \boxplus (gf) = \mu\,((\alpha \boxplus f) \boxplus g)$

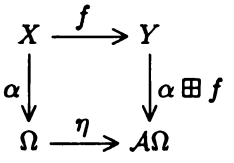
The set Ω^X supports several important operations, such as taking unions and intersections of subsets. Coarse versions of some of these operations can be described. If we denote the meet and join in Ω by $\wedge$ and $\vee$ respectively, unions and intersections are obtained by extending these operations to Ω^X. The corresponding coarse operations in $(\mathcal{A}\Omega)^K$ are defined using the meet and join for the $\leq_t$ ordering. The intuition that these operations in $(\mathcal{A}\Omega)^K$ represent coarsened versions of the corresponding ones in Ω^X is justified by the next result.

Lemma 5

$$\begin{array}{lll} (\alpha \wedge \beta) \boxplus f & \leq_c & (\alpha \boxplus f) \wedge (\beta \boxplus f) \\ (\alpha \vee \beta) \boxplus f & \leq_c & (\alpha \boxplus f) \vee (\beta \boxplus f) \end{array}$$

4 SIMPLE APPROACH TO COARSE REGIONS

We now extend our theory of coarsening sets to the coasening of discrete spatial representations. In terms of the diagram,



we want to consider the situation where X and Y are not merely sets but discrete spaces, and where α is no longer a subset but a subspace of X. To accomplish this extension it is necessary to decide

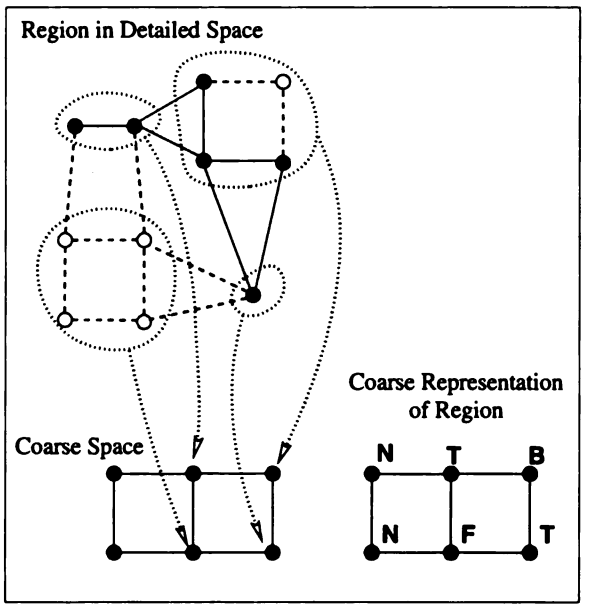
- what is meant by a coarsening of the space X. This should be some analogue of the function $f : X \to Y$.
- what structure Ω of truth values is appropriate for classifying subspaces of X,
- how the structure Ω is modified to yield a structure $\mathcal{A}\Omega$ suitable for classifying coarsened regions.

The answer to the first of these questions is evident:

Definition 3 *A coarsening of discrete spaces $f : (G, \mathrm{adj}_G) \to (F, \mathrm{adj}_F)$ is a homomorphism of the adjacency relations. That is, f is a function from the set G to the set F, such that for all $g_1, g_2 \in G$, $\mathrm{adj}_G(g_1, g_2)$ implies $\mathrm{adj}_F(fg_1, fg_2)$.*

The other two questions can be answered in at least two ways. The simplest possibility is just to take Ω to be the set of two classical truth values, and to represent a region in the space (G, adj_G) by a function from the set of cells, G, to Ω. This allows $\mathcal{A}\Omega$ to be defined as in the previous section. This simple approach has some justification since a region (but not an arbitrary subspace) is specified within a given discrete space by

giving just the set of cells in the region. An example of the representation of a region in a space subjected to a coarsening appears in the following diagram.



This simple approach is adequate for some purposes. It leads to a definition of a coarse region being a labelling of the cells in a discrete space by elements of the set $\{T, B, N, F\}$. However, the adjacency of cells in such a coarse region depends only on the adjacencies in the coarse space, and takes no account of what adjacencies occur between cells in the detailed region.

5 CLASSIFYING FINE REGIONS

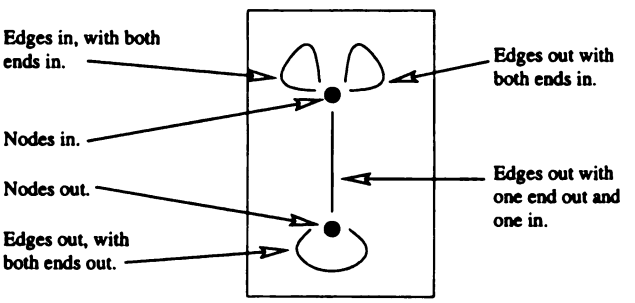
The failure of the simple approach in the previous section to take account of adjacencies can be remedied by taking coarse regions to be spaces having diagrams in which both the edges and nodes are labelled by suitable truth values. To do this it is appropriate to regard a discrete space as an undirected graph. Regions and subspaces can then be defined as graph morphisms from the discrete space to a suitable graph of truth values which plays the role of Ω in the earlier theory of coarse sets.

5.1 CLASSIFIERS

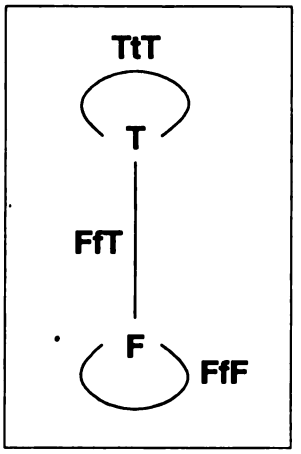
To select a subset of a set X is to classify each element of X as either 'in the subset' or 'not in the subset'. Thus subsets of X are often identified with functions from X to the set of two Boolean truth values. The subsets obtained in this fashion are crisp subsets, but various kinds of fuzzy subset are obtained by considering functions from X to a suitable set of fuzzy truth values.

The general idea of identifying parts of a structure, X, with functions from X to another structure is well-known and leads to the concept of a sub-object classifier in category theory (Goldblatt 1984). The details of this technique applied to crisp subgraphs can be found in (Lawvere & Schanuel 1997), but the details of the approach to coarse graphs given below have not appeared before. There is some material on classifiers for vague graphs and for a simpler kind of coarse graph in (Stell 1999*b*).

To select a subgraph of a graph G is to classify each edge and each node of G as either included in the subgraph or excluded. However, we cannot include an edge without including both its ends. Subgraphs of G correspond to graph morphisms from G to a graph with two nodes, and four edges. The morphism will send each node in G to one of the two nodes in the classifier. One of these nodes receives all the nodes of G which are present in the subgraph, and the other node receives all those nodes of G not present in the subgraph. The classifier has four edges, each one corresponding to one of four categories into which an edge of G can fall. Only one of the edges in the classifier receives edges of G present in the subgraph. The other three deal with edges which are not in the subgraph, but have differing combinations of their endpoints in the subgraph. This classifying graph is shown below, annotated to explain the significance of its nodes and edges.



In a region, rather than an arbitrary subgraph, two nodes which are adjacent in the universal space must be adjacent in the region. This means that any edge with both its ends in the region is automatically itself in the region. Thus regarding a graph G as a discrete space, the regions are identified with morphisms from G to the following graph, which will be denoted Ω_{Reg}.

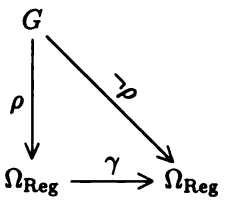


In this graph there are two nodes labelled T and F. These serve to receive the nodes in G which are respectively included in and excluded from the region. The edges in the classifying graph have labels made up of three letters. The first and third letters indicating the two ends of the edge, and the central letter being t or f according as the edge is a target for edges of G present or absent from the region. Thus, for example, the edge FfT is where the edges of G get sent which are not themselves in the region, but which have just one end included in the region.

5.2 DESCRIBING NEGATION

The operations on regions described in section 2 can be described in terms of corresponding operations on classifiers. Here we illustrate the technique in the case of the negation operation $\neg$. The reason for discussing this method is that it provides a guide to defining an analogous operation on coarse regions, and hence to making definitions of the topological style operations which were discussed in section 2.

The classifier $\Omega_{\mathbf{Reg}}$ has only one non-trivial automorphism. This interchanges the two nodes, interchanges the nodes TtT and FfF, and fixes the remaining edge. Denoting this automorphism by γ, the negation of any region $\rho : G \to \Omega_{\mathbf{Reg}}$ is obtained by composing ρ with γ.



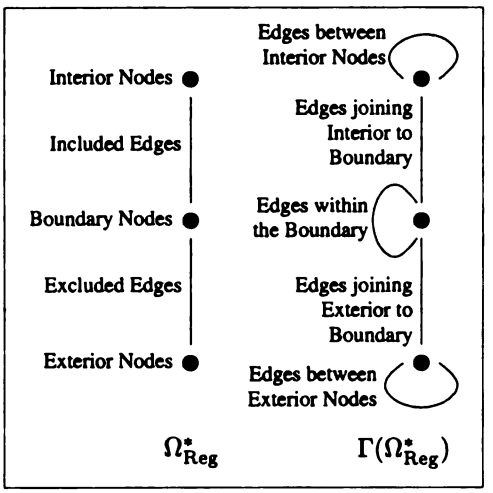
5.3 DESCRIBING SUPPLEMENT

The above treatment of negation arising from the automorphism $\Omega_{\mathbf{Reg}}$ is already well known, but there does not seem to have been an analogous description of the supplement operation $\sim$ before. It is clear that this operation cannot be described in terms of an automorphism of $\Omega_{\mathbf{Reg}}$. However there is an alternative way of classifying regions which does support the required automorphism.

The operations $\neg$ and $\sim$ are informally dual in the sense that the definition of $\neg G$ starts by taking the complement of the set of nodes in G and then adding in certain edges determined by these, whereas the construction of $\sim G$ starts by taking the complement of the set of edges of G and then adding in certain nodes determined by these. To understand the relationship between nodes and edges it is necessary to introduce hypergraphs (Berge 1989). Informally a hypergraph is a generalization of the notion of graph where an edge may be incident with more than two edges.

A hypergraph can be defined as a function $h : E \to \mathcal{P}N$, where E and N are sets of edges and nodes and $\mathcal{P}N$ is the powerset of N. Each hypergraph has dual $h^* : N \to \mathcal{P}E$ where $h^*n = \{e \in E \mid n \in he\}$. A morphism of hypergraphs is a pair of functions $\varphi_N : N_1 \to N_2$ and $\varphi_E : E_1 \to E_2$ such that $\mathcal{P}(\varphi_N)h_1 \subseteq h_2\varphi_E$. If H and K are graphs a hypergraph morphism from H to K may not be a graph morphism since a loop can be mapped to an edge which is not a loop. However, given any hypergraph h we can define a graph having the same nodes as h but with an edge joining node x to y for every edge in h incident with both x and y. If this graph is denoted by $\Gamma(h)$, hypergraph morphisms from a graph K to a hypergraph h correspond to graph morphisms from K to $\Gamma(h)$.

The classifier $\Omega_{\mathbf{Reg}}$ has two nodes and three edges, so its hypergraph dual $\Omega^*_{\mathbf{Reg}}$ has two edges and three nodes. The dual structure is illustrated below, together with the interpretation of the nodes and edges. Note that this interpretation can be constructed from the interpretation of $\Omega_{\mathbf{Reg}}$ by interchanging the concepts of edges and nodes, and also the concepts of included and excluded, so that for instance the edge in $\Omega_{\mathbf{Reg}}$ interpreted as *edges out which are incident with some nodes which are in and some which are out* becomes *nodes in which are incident with some edges in and some edges out.* This latter set of nodes is just the set of boundary nodes.

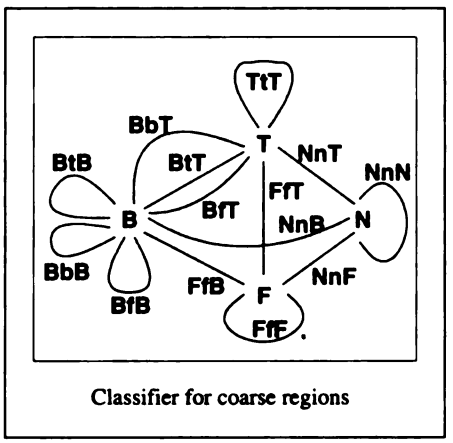


The graph $\Gamma(\Omega^*_{\mathbf{Reg}})$ supports an automorphism, denoted by δ, which corresponds to the symmetry about the horizontal axis in the above diagram. Any region ρ of a discrete space G can be represented as a graph morphism $\rho : G \to \Gamma(\Omega^*_{\mathbf{Reg}})$. The supplement of the region is then obtained by composing with the automorphism δ. Actually this description of supplement only works if we make an additional assumption about our regions. It is necessary to assume they have no isolated nodes, that is nodes included in the region which are adjacent to no other nodes in the region but which are adjacent to other nodes in the universal discrete space. It is possible to remove this assumption, but this requires a different hypergraph from $\Omega^*_{\mathbf{Reg}}$.

6 CLASSIFYING COARSE REGIONS

The general theory of coarsening for sets presented in section 3 can be extended to deal with graphs. A graph of truth values, Ω, is defined to be a lattice of truth values which also carries the structure of an undirected graph. The nodes of the graph are the truth values in the lattice, and every edge in the graph is labelled by some truth value. The same truth value may label more than one edge, but distinct edges with the same ends must carry distinct labels. The construction of $\mathcal{A}\Omega$ from Ω is made as before at the level of nodes. For the edges, if $[a, b]$ and $[c, d]$ are two nodes in $\mathcal{A}\Omega$ then there is an edge between $[a, b]$ and $[c, d]$ labelled by $[p, q]$ iff there are nodes in Ω: $w, x \in [a, b]$ and $y, z \in [c, d]$ and edges in Ω between w and y and between x and z labelled by p and q. In addition, there is an edge labelled by $[\top, \bot]$ (the empty interval) joining $[\top, \bot]$ to each node of $\mathcal{A}\Omega$.

Applying this construction to the classifying graph $\Omega_{\mathbf{Reg}}$ leads to the following graph, $\mathcal{A}\Omega_{\mathbf{Reg}}$.



Classifier for coarse regions

The nodes in this graph are labelled by the four truth values, T, F, N, B which were introduced in section 3 above. The edges have been labelled by three values, as in the labelling of $\Omega_{\mathbf{Reg}}$. Thus for example, the edge labelled BtT is where all edges of the universal region are mapped which have one end node with value B, the other end node T, and where the edge itself has the value true (i.e. is included in the region).

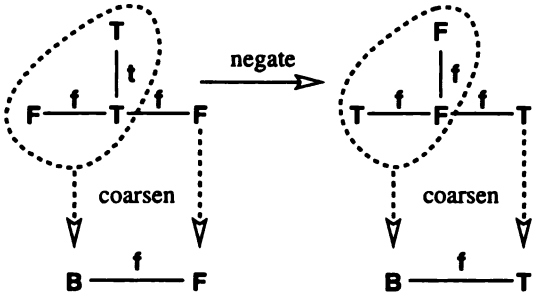
6.1 NEGATION FOR COARSE REGIONS

We saw in section 5 that a region, within a universal discrete space F, could be represented as a graph morphism $\rho : F \to \Omega_{\mathbf{Reg}}$, and that the negation of ρ was obtained by composing ρ with the automorphism of $\Omega_{\mathbf{Reg}}$ denoted by γ. This operation of composing with γ gives us a function $\gamma^F : \Omega^F_{\mathbf{Reg}} \to \Omega^F_{\mathbf{Reg}}$, where $\Omega^F_{\mathbf{Reg}}$ denotes the set of all graph morphisms from F to $\Omega_{\mathbf{Reg}}$ i.e. the set of all regions within F. In this section this construction is generalized to the coarse case by using a graph morphism $\eta : \mathcal{A}\Omega_{\mathbf{Reg}} \to \mathcal{A}\Omega_{\mathbf{Reg}}$, which plays the same role in the coarse case as γ does in the non-coarse case. The relationship between the two forms of negation and coarsening is summarized by the following diagram.

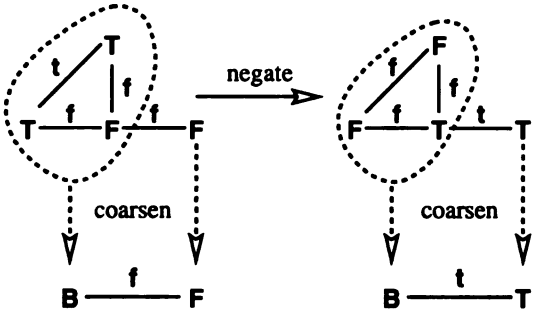
$$\begin{array}{ccc} \Omega^F_{\mathbf{Reg}} & \xrightarrow{\_\boxplus f} & (\mathcal{A}\Omega_{\mathbf{Reg}})^G \\ \Big\downarrow{\scriptstyle \gamma^F} & \leq_c & \Big\downarrow{\scriptstyle \eta^G} \\ \Omega^F_{\mathbf{Reg}} & \xrightarrow[\_\boxplus f]{} & (\mathcal{A}\Omega_{\mathbf{Reg}})^G \end{array}$$

The effect of η on nodes is to fix B and N, and to interchange T and F. On many of the edges, the effect is determined by the action on the nodes. The other cases are that all the loops on B are sent to BbB, and that FfB is sent to BbT. To see that this last case is correct, consider the following two examples. Each

example considers what can happen when a region is negated then coarsened, as opposed to coarsening the region first.



In the second example the region has exactly the same coarse representation, but a different negation.



7 CONCLUSIONS AND FURTHER WORK

This paper has introduced an algebraic approach to the representation of, and the operations on, discrete spaces. It has also provided the beginnings of a formal theory of coarsening applied to discrete space. This formal theory is not yet fully developed, and further work is necessary in a number of directions. In particular, relationships between coarse discrete regions, analogous to the RCC5 and RCC8 schemes, will be investigated. Another area for further work is the development of an implementation of the operations on coarse discrete regions. This would raise questions such as the complexity of reasoning with coarse discrete spaces. Analogous questions have already been studied in the non-discrete case (Renz & Nebel 1999, Papadimitriou, Suciu & Vianu 1999).

Acknowledgements

I am grateful to the two referees for some useful comments. The use of Paul Taylor's macros for the category theoretic diagrams is acknowledged. The work is supported by EPSRC under grant GR/M 56685 'Managing Vagueness, Uncertainty, and Granularity in Spatial Information Systems'.

References

Arieli, O. & Avron, A. (1996), 'Reasoning with logical bilattices', *Journal of Logic, Language and Information* **5**, 25–63.

Arieli, O. & Avron, A. (1998), 'The value of the four values', *Artificial Intelligence* **102**, 97–141.

Asher, N. & Vieu, L. (1995), Toward a geometry of common sense: A semantics and a complete axiomatisation of mereotopology, *in* '14th International Joint Conference on Artificial Intelligence, IJCAI'95', pp. 846–852.

Belnap, N. D. (1977), A useful four-valued logic, *in* G. Epstein & J. M. Dunn, eds, 'Modern Uses of Multiple-Valued Logic', D. Reidel, pp. 8–37.

Berge, C. (1989), *Hypergraphs: Combinatorics of Finite Sets*, Vol. 45 of *North-Holland Mathematical Library*, North-Holland.

Bittner, T. & Stell, J. G. (1998), 'A boundary-sensitive approach to qualitative location', *Annals of Mathematics and Artificial Intelligence* **24**, 93–114.

Borgo, S., Guarino, N. & Masolo, C. (1996), A pointless theory of space based on strong connection and congruence, *in* L. Aiello, J. Doyle & S. C. Shapiro, eds, 'Principles of Knowledge Representation and Reasoning. Proceedings of the Fifth International Conference', pp. 220–229.

Casati, R. & Varzi, A. C. (1999), *Parts and Places. The Structures of Spatial Representation*, MIT Press.

Cohn, A. G., Bennett, B. et al. (1997), 'Qualitative spatial representation and reasoning with the region connection calculus', *GeoInformatica* **1**, 275–316.

Egenhofer, M. J. & Franzosa, R. (1991), 'Point-set topological spatial relations', *International Journal of Geographical Information Systems* **5**, 161–174.

Egenhofer, M. J. & Franzosa, R. (1995), 'On the equivalence of topological relations', *International Journal of Geographical Information Systems* **9**, 133–152.

Euzenat, J. (1994), Granularié dans les représentations spatio-temporelles, Rapport de recherche 2242, INRIA Rhone-Alpes.

Euzenat, J. (1995), An algebraic approach to granularity in qualitative time and space representation, *in* '14th International Joint Conference on Artificial Intelligence, IJCAI'95', pp. 894–900.

Galton, A. (1998), 'Modes of overlap', *Journal of Visual Languages and Computing* **9**, 61–79.

Galton, A. (1999), The mereotopology of discrete space, *in* C. Freksa & D. Mark, eds, 'Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science. International Conference COSIT'99', Vol. 1661 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 251–266.

Ginsberg, M. L. (1988), 'Multivalued logics: A uniform approach to reasoning in AI', *Computer Intelligence* **4**, 256–316.

Goldblatt, R. (1984), *Topoi. The Categorial Analysis of Logic*, North-Holland.

Hirtle, S. C. & Frank, A. U., eds (1997), *Spatial Information Theory, International Conference COSIT'97, Proceedings*, Vol. 1329 of *Lecture Notes in Computer Science*, Springer-Verlag.

Kaufman, S. G. (1991), A formal theory of spatial reasoning, *in* J. Allen, R. Fikes & E. Sandewall, eds, 'Principles of Knowledge Representation and Reasoning. Proceedings of the Second International Conference (KR91)', Morgan Kaufmann, pp. 347–356.

Lawvere, F. W. (1986), Introduction, *in* F. W. Lawvere & S. H. Schanuel, eds, 'Categories in Continuum Physics', Vol. 1174 of *Lecture Notes in Mathematics*, Springer-Verlag, pp. 1–16.

Lawvere, F. W. & Schanuel, S. H. (1997), *Conceptual Mathematics. A First Introduction to Categories*, Cambridge University Press.

Masolo, C. & Vieu, L. (1999), Atomicity vs infinite divisibility of space, *in* C. Freksa & D. Mark, eds, 'Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science. International Conference COSIT'99', Vol. 1661 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 235–250.

Müller, J. C., Lagrange, J. P. & Weibel, R., eds (1995), *GIS and Generalisation: Methodology and Practice*, Taylor and Francis, London.

Orłowska, E., ed. (1998), *Incomplete Information - Rough Set Analysis*, Vol. 13 of *Studies in Fuzziness and Soft Computing*, Physica-Verlag, Heidelberg.

Papadimitriou, C. H., Suciu, D. & Vianu, V. (1999), 'Topological queries in spatial databases', *Journal of Computer and System Sciences* **58**(1), 29–53.

Randall, D. A. & Cohn, A. G. (1989), Modelling topological and metrical properties in physical processes, *in* R. J. Brachman, H. J. Levesque & R. Reiter, eds, 'Proceedings of First International Conference on Principles of Knowledge Representation and Reasoning', Morgan Kaufmann, pp. 357–368.

Randall, D. A., Cui, Z. & Cohn, A. G. (1992), A spatial logic based on regions and connection, *in* B. Nebel, C. Rich & W. Swartout, eds, 'Principles of Knowledge Representation and Reasoning. Proceedings of the Third International Conference (KR92)', Morgan Kaufmann, pp. 165–176.

Renz, J. & Nebel, B. (1999), 'On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus', *Artificial Intelligence* **108**(1–2), 69–123.

Reyes, G. E. & Zolfaghari, H. (1996), 'Bi-Heyting algebras, toposes and modalities', *Journal of Philosophical Logic* **25**, 25–43.

Smith, B. (1996), 'Mereotopology - A theory of parts and boundaries', *Data and Knowledge Engineering* **20**, 287–303.

Smith, B. (1997), Boundaries: An essay in meretopology, *in* L. E. Hahn, ed., 'The Philosophy of Roderick M. Chisholm', Open Court Publishing Co., pp. 532–561.

Stell, J. G. (1999*a*), 'Boolean connection algebras: A new approach to the region-connection calculus', Submitted for Publication. `www.keele.ac.uk/depts/cs/Staff/Homes/John/papers/index.html`.

Stell, J. G. (1999*b*), Granulation for graphs, *in* C. Freksa & D. Mark, eds, 'Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science. International Conference COSIT'99', Vol. 1661 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 417–432.

Stell, J. G. & Worboys, M. F. (1997), The algebraic structure of sets of regions, *in* Hirtle & Frank (1997), pp. 163–174.

Stell, J. G. & Worboys, M. F. (1998), Stratified map spaces: A formal basis for multi-resolution spatial databases, *in* T. K. Poiker & N. Chrisman, eds, 'SDH'98 Proceedings 8th International Symposium on Spatial Data Handling', International Geographical Union, pp. 180–189.

Stell, J. G. & Worboys, M. F. (1999), Generalizing graphs using amalgamation and selection, *in* R. H. Güting, D. Papadias & F. Lochovosky, eds, 'Advances in Spatial Databases. 6th International Symposium, SSD'99', Vol. 1651 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 19–32.

Timpf, S. & Frank, A. U. (1997), Using hierarchical spatial data structures for hierarchical spatial reasoning, *in* Hirtle & Frank (1997), pp. 69–83.

van Lambalgen, M. & van der Does, J. (1997), A logic of vision, Technical Report LP-07-1997-07, Institute for Logic, Language and Computation, University of Amsterdam, `www.illc.uva.nl/Publications/index.html`.

Worboys, M. F. (1998), 'Imprecision in finite resolution spatial data', *GeoInformatica* **2**, 257–279.

Nonmonotonic Reasoning I

Finding Admissible and Preferred Arguments Can be Very Hard

Yannis Dimopoulos
University of Cyprus
Dept. of Computer Science
75 Kallipoleos Street
Nicosia P.O. Box 537
Cyprus

Bernhard Nebel
Universität Freiburg
Institut für Informatik
Am Flughafen 17
D-79110 Freiburg
Germany

Francesca Toni
Imperial College
Dept. of Computing
180 Queen's Gate
London SW7 2BZ
United Kingdom

Abstract

Bondarenko *et al.* have recently proposed an extension of the argumentation-theoretic semantics of *admissible* and *preferred arguments*, originally proposed for logic programming only, to a number of other nonmonotonic reasoning formalisms. In this paper we analyse the computational complexity of *credulous* and *sceptical* reasoning under the semantics of admissible and preferred arguments for (the propositional variant of) some well-known frameworks for nonmonotonic reasoning, i.e. Theorist, Circumscription and Autoepistemic Logic. While the new semantics have been assumed to mitigate the computational problems of nonmonotonic reasoning under the standard semantics of *stable extensions*, we show that in many cases reasoning under the new semantics is computationally harder than under the standard semantics. In particular, for Autoepistemic Logic, the sceptical reasoning problem under the semantics of preferred arguments is located at the fourth level of the polynomial hierarchy, two levels above the same problem under the standard semantics. In some cases, however, reasoning under the new semantics becomes easier – reducing to reasoning in the monotonic logics underlying the nonmonotonic frameworks.

1 Introduction

Bondarenko *et al.* [1997] propose a single abstract framework for nonmonotonic reasoning that can be instantiated to capture many existing logics for nonmonotonic reasoning, in particular Theorist [Poole, 1988], Circumscription [McCarthy, 1980], and Autoepistemic Logic (AEL) [Moore, 1985]. They also propose two new (argumentation-theoretic) semantics for nonmonotonic reasoning, generalising the admissibility semantics [Dung, 1991] and the semantics of preferred extensions [Dung, 1991] or partial stable models [Saccà and Zaniolo, 1990] for logic programming. In this paper we refer to the new semantics as *admissible* and *preferred arguments*, respectively.

The new semantics are more general than the standard semantics of *stable extensions* for nonmonotonic reasoning, since every stable extension is a preferred (and admissible) argument, but not every preferred argument is a stable extension. Moreover, the new semantics are more liberal because for most concrete logics for nonmonotonic reasoning, admissible and preferred arguments are always guaranteed to exist, whereas stable extensions are not. Finally, reasoning under the new semantics appears to be computationally easier than reasoning under the standard semantics [Kowalski and Toni, 1996; Dung *et al.*, 1997].

However, from a complexity-theoretic point of view, it seems unlikely that the new semantics lead to better lower bounds than the standard semantics since all the "sources of complexity" one has in nonmonotonic reasoning are still present under the new semantics. There are potentially exponentially many arguments sanctioned by the semantics. Further, in order to test whether a sentence is entailed by a particular argument one has to reason in the underlying monotonic logic. For this reason, one would expect that reasoning under the new semantics has the same complexity as under the standard semantics, i.e., it is on the second level of the polynomial hierarchy for frameworks with full propositional logic as the underlying logic [Cadoli and Schaerf, 1993]. However, recent results for disjunctive logic programming [Eiter *et al.*, 1998] and default logic [Dimopoulos *et al.*, 1999] show that reasoning under the semantics of preferred extensions

can be harder than under the standard semantics, i.e., it is on the third level of the polynomial hierarchy.

In this paper we extend this analysis and provide complexity results for reasoning in the *propositional variants* of Theorist Circumscription, and AEL under the new semantics. In particular, we show that, for AEL, credulous and sceptical reasoning under the admissibility semantics is on the third level of the polynomial hierarchy and sceptical reasoning under the preferability semantics is on the fourth level of the polynomial hierarchy.

The paper is organised as follows. Section 2 summarises relevant features of the abstract framework of [Bondarenko *et al.*, 1997], its semantics and concrete instances. Section 3 introduces the reasoning problems and gives generic upper bounds for credulous and sceptical reasoning, parametric wrt the complexity of the underlying monotonic logics. Section 4 gives then completeness results for Theorist and Circumscription, and Section 5 provides the completeness results for AEL. Section 6 discusses the results and concludes.

2 Nonmonotonic Reasoning via Argumentation

Assume a **deductive systems** $(\mathcal{L}, \mathcal{R})$, where $\mathcal{L}$ is some formal language with countably many sentences and $\mathcal{R}$ is a set of inference rules inducing a monotonic derivability notion $\vdash$. Given a theory $T \subseteq \mathcal{L}$ and a formula $\alpha \in \mathcal{L}$, $Th(T) = \{\alpha \in \mathcal{L} \mid T \vdash \alpha\}$ is the deductive closure of T. Then, an **abstract (assumption-based) framework** is a triple $\langle T, A, \overline{\ }\rangle$, where $T, A \subseteq \mathcal{L}$ and $\overline{\ }$ is a mapping from A into $\mathcal{L}$. T, the **theory**, is a (possibly incomplete) set of beliefs, formulated in the underlying language, and can be extended by subsets of A, the set of **assumptions**. Indeed, an **extension** of an abstract framework $\langle T, A, \overline{\ }\rangle$ is a theory $Th(T \cup \Delta)$, with $\Delta \subseteq A$ (sometimes an extension is referred to simply as $T \cup \Delta$ or Δ). Finally, given an assumption $\alpha \in A$, $\overline{\alpha}$ denotes the **contrary** of α.

Theorist [Poole, 1988] can be understood as a framework $\langle T, A, \overline{\ }\rangle$ where T and A are both arbitrary sets of sentences of classical (first-order or propositional) logic and the contrary $\overline{\alpha}$ of an assumption α is just its negation $\neg\alpha$. $\vdash$ is ordinary classical provability.

Circumscription [McCarthy, 1980] can be understood similarly, except that the assumptions are negations of atomic sentences $\neg p(t)$, for all predicates p which are minimised, and atomic sentences $q(t)$ or their negation $\neg q(t)$, for all predicates q which are fixed.

Autoepistemic logic (AEL) [Moore, 1985] has as the underlying language $\mathcal{L}$ a modal logic with a modal operator L, but the inference rules in $\mathcal{R}$ are those of classical logic. The assumptions have the form $\neg L\alpha$ or $L\alpha$. The contrary of $\neg L\alpha$ is α, and the contrary of $L\alpha$ is $\neg L\alpha$.

Given an abstract framework $\langle T, A, \overline{\ }\rangle$ and an assumption set $\Delta \subseteq A$, Δ **attacks an assumption** $\alpha \in A$ iff $\overline{\alpha} \in Th(T \cup \Delta)$, and Δ **attacks an assumption set** $\Delta' \subseteq A$ iff Δ attacks some assumption $\alpha \in \Delta'$.

Given that an assumption set $\Delta \subseteq A$ is **closed** iff $\Delta = A \cap Th(T \cup \Delta)$, the standard semantics of extensions of Theorist [Poole, 1988] and stable expansions of AEL [Moore, 1985] correspond to the "stability" semantics of abstract frameworks, where an assumption set $\Delta \subseteq A$ is **stable** iff

1. Δ is closed,
2. Δ does not attack itself, and
3. Δ attacks each assumption $\alpha \notin \Delta$.

A **stable extension** is an extension $Th(T \cup \Delta)$ for some stable assumption set Δ. The standard semantics of Circumscription [McCarthy, 1980] corresponds to the intersection of all stable extensions of the abstract framework corresponding to Circumscription.

Bondarenko *et al.* [1997] argue that the stability semantics is unnecessarily restrictive, because it insists that an assumption set should take a stand on every issue (assumption). Thus, they define new semantics for the abstract framework by generalising the "admissibility" semantics originally proposed for logic programming by Dung [1991]. An assumption set $\Delta \subseteq A$ is **admissible** iff

1. Δ is closed,
2. Δ does not attack itself, and
3. for all closed assumption sets $\Delta' \subseteq A$, if Δ' attacks Δ then Δ attacks Δ'.

Maximal (wrt set inclusion) admissible assumption sets are called **preferred**. In this paper we use the following terminology: an **admissible (preferred) argument** is an extension $Th(T \cup \Delta)$ for some admissible (preferred) assumption set Δ. Bondarenko *et al.* show that, in the concrete instance of the abstract framework corresponding to logic programming, preferred arguments correspond to preferred extensions [Dung, 1991] and partial stable models [Saccà and Zaniolo, 1990].

Every stable assumption set/extension is preferred (and thus admissible) [Bondarenko *et al.*, 1997, Theorem 4.6], but not vice versa, in general. However, if the framework is **normal**, i.e., if every maximal closed set that does not attack itself is a stable set, then the preferability and stability semantics coincide [Bondarenko *et al.*, 1997, Theorem 4.8]. Examples of normal frameworks are Theorist and Circumscription.

In order to illustrate the effects of the different semantics, let us consider the following AEL theory:

$$\neg Lp \rightarrow p; \qquad \neg Lq \rightarrow r.$$

This theory has no stable extension, one preferred argument $\{\neg Lq, Lr, L\neg Lq, \ldots\}$, and a number of admissible arguments, e.g., in addition to the preferred argument, the arguments $\emptyset$, $\{\neg Lq\}$, $\{\neg Lq, Lr\}$, $\{\neg Lq, L\neg Lq\}$. If we drop the sentence "$\neg Lp \rightarrow p$," we get the same admissible and preferred arguments. In addition, the preferred argument is also stable.

In the sequel, we will often use the following property [Bondarenko *et al.*, 1997, Theorem 4.8]: Every preferred assumption set is admissible and every admissible assumption set is a subset of some preferred assumption set.

3 Reasoning Problems and Generic Upper Bounds

We will analyse the *computational complexity* of the following reasoning problems for the *propositional variants* of the frameworks for Theorist, Circumscription and AEL under admissibility and preferability semantics:

- the **credulous reasoning problem**, i.e., the problem of deciding for any given sentence φ whether $\varphi \in Th(T \cup \Delta)$ for *some* assumption set Δ sanctioned by the semantics;
- the **sceptical reasoning problem**, i.e., the problem of deciding for any given sentence φ whether $\varphi \in Th(T \cup \Delta)$ for *all* assumption sets Δ sanctioned by the semantics.

The sentence φ is any (variable-free) modal sentence in the AEL case, and any formula in propositional logic in the Theorist and Circumscription cases.

Instead of the sceptical reasoning problem, we will often consider its complementary problem, i.e.

- **the co-sceptical reasoning problem**, i.e, the problem of deciding for any given sentence φ whether $\varphi \notin Th(T \cup \Delta)$ for *some* assumption set Δ sanctioned by the semantics.

In addition, we will consider a sub-problem of all these problems, namely:

- the **assumption set verification problem**, i.e., the problem of deciding whether a given assumption set Δ is sanctioned by the semantics.

The computational complexity [1] of the above problems for all frameworks and semantics we consider is located at the lower end of the *polynomial hierarchy*. This is an infinite hierarchy of complexity classes above NP defined by using *oracle machines*, i.e. Turing machines that are allowed to call a subroutine—the *oracle*—deciding some fixed problem in constant time. Let $\mathcal{C}$ be a class of decision problems. Then, for any class $\mathcal{X}$ defined by resource bounds, $\mathcal{X}^{\mathcal{C}}$ denotes the class of problems decidable on a Turing machine with an oracle for a problem in $\mathcal{C}$ and a resource bound given by $\mathcal{X}$. Based on these notions, the sets Δ_k^p, Σ_k^p, and Π_k^p are defined as follows:

$$\Delta_0^p = \Sigma_0^p = \Pi_0^p = \mathsf{P}$$

$$\Delta_{k+1}^p = \mathsf{P}^{\Sigma_k^p}, \quad \Sigma_{k+1}^p = \mathsf{NP}^{\Sigma_k^p}, \quad \Pi_{k+1}^p = \text{co-}\mathsf{NP}^{\Sigma_k^p}.$$

The "canonical" complete problems are SAT for Σ_1^p=NP and k-QBF for Σ_k^p ($k > 1$), where k-QBF is the problem of deciding whether the quantified boolean formula

$$\underbrace{\exists \vec{p} \forall \vec{q} \ldots}_{k \text{ alternating quantifiers starting with } \exists} \Phi(\vec{p}, \vec{q}, \ldots).$$

is true. The complementary problem, denoted by co-k-QBF, is complete for Π_k^p.

All problems in the polynomial hierarchy can be solved in polynomial time iff P = NP. Further, all these problems can be solved by worst-case exponential time algorithms. Thus, the polynomial hierarchy might not seem too meaningful. However, different levels of the hierarchy differ considerably in practice, e.g. methods working for moderately sized instances of NP-complete problems do not work for Σ_2^p-complete problems.

The complexity of the problems we are interested in has been extensively studied for existing logics for nonmonotonic reasoning under the standard, stability

[1] For the following, we assume that the reader is familiar with the basic concepts of complexity theory, i.e., the complexity classes P, NP and co-NP, and the notion of completeness with respect to log-space reductions. Good textbooks covering these notions are [Garey and Johnson, 1979] and [Papadimitriou, 1994].

semantics [Cadoli and Schaerf, 1993; Gottlob, 1992; Niemelä, 1990; Marek and Truszczynski, 1993]. In particular, the credulous and sceptical reasoning problems are located at the second level of the polynomial hierarchy for the formalisms we are interested in, i.e., sceptical reasoning is Π_2^p-complete and credulous reasoning is Σ_2^p-complete for all of Theorist, Circumscription and AEL.

In this section we prove a number of *generic* upper bounds for reasoning under the admissibility and preferability semantics that are parametric on the complexity of the *derivability problem* in the underlying monotonic logic. For all of Theorist, Circumscription and AEL, the underlying logic is classical propositional logic, hence the derivability problem is in co-NP.

In order to decide the credulous and co-sceptical reasoning problems, one can apply the following "*guess & check*" algorithm:

(i) guess an assumption set,

(ii) verify that it is sanctioned by the semantics, and

(iii) verify that the formula under consideration is derivable from the assumption set and the monotonic theory or not derivable from it, respectively.

¿From that it follows that credulous reasoning and co-sceptical reasoning are in the complexity class $\mathsf{NP}^{\mathcal{C}}$, provided reasoning in the underlying logic is in $\mathcal{C}$ and the verification that an assumption set is sanctioned by the semantics can be done with polynomially many calls to a $\mathcal{C}$-oracle. For the stability semantics, we need indeed only polynomially many $\mathcal{C}$-oracle calls in order to verify that the assumption set Δ is not self-attacking and that it is closed and attacks all assumptions $\alpha \notin \Delta$. For the admissibility and preferability semantics, instead, the verification step does not appear to be that easy.

Theorem 1 *For frameworks with an underlying monotonic logic with a derivability problem in $\mathcal{C}$, the* assumption set verification problem *is*

1. *in $\mathsf{P}^{\mathcal{C}}$ under the stability semantics,*
2. *in co-$\mathsf{NP}^{\mathcal{C}}$ under the admissibility semantics,*
3. *in co-$\mathsf{NP}^{\mathsf{NP}^{\mathcal{C}}}$ under the preferability semantics.*

Proof: Claim (1) follows from the argument above that polynomially many $\mathcal{C}$-oracle calls are sufficient to verify that an assumption set is stable.

In order to prove claim (2), we specify the following nondeterministic, polynomial-time algorithm that uses a $\mathcal{C}$-oracle:

1. Check if Δ is closed, using $|A - \Delta|$ $\mathcal{C}$-oracle calls. If not, succeed.
2. Check if Δ is self-attacking using $|\Delta|$ $\mathcal{C}$-oracle calls. If so, succeed.
3. Guess a set $\Delta' \subseteq A$.
4. Verify that Δ' is closed, using $|A - \Delta'|$ $\mathcal{C}$-oracle calls.
5. Verify that Δ' attacks Δ using $|\Delta|$ $\mathcal{C}$-oracle calls.
6. Verify that Δ does not attack Δ' using $|\Delta'|$ $\mathcal{C}$-oracle calls.

Obviously, the algorithm succeeds iff Δ is not admissible, i.e., it solves the complement of the assumption set verification problem.

Claim (3) is proved by the following nondeterministic, polynomial-time algorithm that uses an $\mathsf{NP}^{\mathcal{C}}$-oracle:

1. Check if Δ is admissible, which, by claim (2), can be done by one call to an $\mathsf{NP}^{\mathcal{C}}$-oracle. If it is not, succeed.
2. Guess a set $\Delta' \supset \Delta$.
3. Check if Δ' is admissible. If it is, succeed. Otherwise fail.

Obviously, this algorithm decides the complement of the assumption set verification problem under the preferability semantics, demonstrating that this problem is in co-$\mathsf{NP}^{\mathsf{NP}^{\mathcal{C}}}$. ■

Furthermore, there does not appear to be a way around the difficulty of the assumption set verification problem in the general case. For some special frameworks, however, the problem can be solved more efficiently. In particular, this is the case for normal frameworks, at least under the preferability semantics. Indeed, since preferred assumption sets coincide with stable assumption sets for these frameworks, the result for the stability semantics in Theorem 1 applies.

Corollary 2 *For* normal *frameworks with an underlying monotonic logic with a derivability problem in $\mathcal{C}$, the* assumption set verification problem *under the preferability semantics is in $\mathsf{P}^{\mathcal{C}}$.*

We could now apply the *guess & check* algorithm described above for deriving upper bounds for the reasoning problems. However, the following considerations (sometimes) allow to obtain better upper bounds for the reasoning problems than the ones obtained by using the *guess & check* algorithm in conjunction with Theorem 1.

First, since any preferred assumption set is admissible and any admissible set is a subset of a preferred set, the following result holds.

Proposition 3 *Credulous reasoning under the admissibility semantics is equivalent to credulous reasoning under the preferability semantics.*

Thus, credulous reasoning under the admissibility semantics has the same upper bound as credulous reasoning under the preferability semantics. This implies that, for normal frameworks, credulous reasoning under admissibility and preferability semantics has the same upper bound as credulous reasoning under the stability semantics.

In addition, co-sceptical and sceptical reasoning under the admissibility semantics is often much simpler than suggested by the upper bounds in Theorem 1 for the respective assumption set verification problem. We call an abstract framework $\langle T, A, \overline{\ } \rangle$ **simple** iff there is no admissible set if T is inconsistent and otherwise there exists a *least* admissible set $\Delta_l = A \cap Th(T)$.

Proposition 4 *For* simple *frameworks with an underlying monotonic logic with derivability problem in $\mathcal{C}$, the sceptical reasoning problem under the admissibility semantics is in $\mathcal{C}$.*

Putting all the above results together and applying the *guess & check* algorithm leads to the next theorem, specifying upper bounds for all the frameworks and reasoning problems of interest.

Theorem 5 *Upper bounds for the different reasoning problems are as specified in the following table:*

Upper bounds for credulous reasoning			
	Stability	*Admissibility*	*Preferability*
general	$\mathsf{NP}^{\mathcal{C}}$	$\mathsf{NP}^{\mathsf{NP}^{\mathcal{C}}}$	$\mathsf{NP}^{\mathsf{NP}^{\mathcal{C}}}$
normal	$\mathsf{NP}^{\mathcal{C}}$	$\mathsf{NP}^{\mathcal{C}}$	$\mathsf{NP}^{\mathcal{C}}$
simple	$\mathsf{NP}^{\mathcal{C}}$	$\mathsf{NP}^{\mathsf{NP}^{\mathcal{C}}}$	$\mathsf{NP}^{\mathsf{NP}^{\mathcal{C}}}$
Upper bounds for sceptical reasoning			
	Stability	*Admissibility*	*Preferability*
general	$\text{co-NP}^{\mathcal{C}}$	$\text{co-NP}^{\mathsf{NP}^{\mathcal{C}}}$	$\text{co-NP}^{\mathsf{NP}^{\mathsf{NP}^{\mathcal{C}}}}$
normal	$\text{co-NP}^{\mathcal{C}}$	$\text{co-NP}^{\mathsf{NP}^{\mathcal{C}}}$	$\text{co-NP}^{\mathcal{C}}$
simple	$\text{co-NP}^{\mathcal{C}}$	$\mathcal{C}$	$\text{co-NP}^{\mathsf{NP}^{\mathsf{NP}^{\mathcal{C}}}}$

Proof: The columns for the *stability* semantics follow because the credulous and co-sceptical reasoning problems can be decided by an NP guessing step of an assumption set, followed by the verification that the assumption set is stable, which by Theorem 1 can be solved by a call to a $\mathsf{P}^{\mathcal{C}}$ oracle, followed in turn by the verification that the desired sentence is derivable or not derivable, respectively, from the theory augmented by the guessed assumption set, which can be solved by a call to a $\mathcal{C}$-oracle. This means that $\mathsf{NP}^{\mathsf{P}^{\mathcal{C}}} = \mathsf{NP}^{\mathcal{C}}$ is an upper bound for the credulous and co-sceptical reasoning problems under the stability semantics.

The results for the admissibility semantics in the *general* case follow by the same kind of argument. The entry for credulous reasoning under the preferability semantics in the general case follows from Proposition 3 and the entry for the admissibility semantics in the same table. The entry for sceptical reasoning under the preferability semantics in the general case follows again by using a *guess & check* algorithm and Theorem 1.

The results for *normal* frameworks are justified as follows. Credulous reasoning under the admissibility and preferability semantics can be shown to be in $\mathsf{NP}^{\mathcal{C}}$ by using the *guess & check* algorithm and applying Proposition 3 and Corollary 2. Since the same applies for co-sceptical reasoning under the preferability semantics, membership in $\text{co-NP}^{\mathcal{C}}$ follows. Finally, the upper bound for sceptical reasoning under the admissibility semantics is the same as the upper bound for the same problem in the general case.

Finally, the results for *simple* frameworks are the general results with the exception of sceptical reasoning under the admissibility semantics where the upper bound is the one given by Proposition 4. ■

As shown in Theorem 5 and already remarked above, sceptical reasoning under the admissibility semantics can be sometimes simpler than sceptical reasoning under the stability semantics. However, in all such cases the problem reduces to deriving monotonic conclusions from the theory T alone, ignoring the assumptions completely. In other words, in these cases, sceptical nonmonotonic reasoning reduces to monotonic reasoning and is thus trivialised.

4 Simple, Normal Frameworks: Theorist and Circumscription

The frameworks for Theorist and Circumscription are *normal* (see [Bondarenko *et al.*, 1997]) and *simple*, as shown below.

Lemma 6 *The frameworks for Theorist and Circumscription are simple.*

Proof: Circumscription is a special instance of Theorist. Thus, we only need to prove the theorem for Theorist.

If the given Theorist theory T is inconsistent then the corresponding framework admits no admissible argument, as any closed assumption set attacks itself.

Assume that T is consistent. Then, we only need to prove that $\Delta = Th(T) \cap A$ attacks every closed assumption set Δ' which attacks Δ. Now, if $\Delta = \emptyset$, then there is no set Δ' that attacks Δ. If $\Delta \neq \emptyset$, then Δ' attacks Δ iff $T \cup \Delta'$ is inconsistent and, as Δ' is closed, $\Delta' = A$. Thus, necessarily Δ attacks Δ'. ■

For both frameworks, the credulous and sceptical reasoning problems reach the respective minimal upper bounds specified in Theorem 5. Indeed, because of Proposition 3 and the fact that the frameworks are normal, credulous reasoning under the admissibility and preferability semantics is identical to credulous reasoning under the standard, stability semantics, leading to the result that the complexity of credulous reasoning under all three semantics is identical.

Corollary 7 *Credulous reasoning in Theorist and Circumscription under the admissibility and preferability semantics is Σ_2^p-complete.*

Similarly, for normal frameworks, sceptical reasoning under the preferability semantics is identical to sceptical reasoning under the stability semantics, from which the next result follows immediately.

Corollary 8 *Sceptical reasoning in Theorist and Circumscription under the preferability semantics is Π_2^p-complete.*

Finally, sceptical reasoning under the admissibility semantics is trivial as Theorist and Circumscription are simple frameworks and thus sceptical reasoning reduces to classical derivability.

Corollary 9 *Sceptical reasoning in Theorist and Circumscription under the admissibility semantics is* co-NP*-complete.*

In other words, for Theorist and Circumscription, we either get the same results under the new semantics as under the stability semantics or we get trivial results.

5 General Frameworks: Autoepistemic Logic

AEL is neither normal nor simple. In addition, it satisfies no other property that might help to reduce the computational complexity that is suggested by the upper bound results in Section 3. (In particular, AEL is not *flat*, see [Bondarenko *et al.*, 1997; Dimopoulos *et al.*, 1999]). We prove that these upper bounds are tight for AEL. This implies that, in AEL, even under the admissibility semantics sceptical reasoning is harder than under the standard semantics – a phenomenon we do not have with simple, normal, or flat frameworks.

Theorem 10 *In AEL, credulous reasoning under the admissibility semantics is Σ_3^p-complete and sceptical reasoning under the admissibility semantics is Π_3^p-complete.*

Proof: Membership for credulous and sceptical reasoning follows from Theorem 5.

Hardness for credulous reasoning will be proven by a reduction from 3-QBF. Assume the following quantified boolean formula: $\exists p_1,\ldots,p_n\ \forall q_1,\ldots,q_m\ \exists r_1,\ldots,r_k \Phi$, with Φ a formula in 3CNF over the propositional variables $p_1,\ldots,p_n$, $q_1,\ldots,q_m$, $r_1,\ldots,r_k$. We construct an AEL theory T such that the given quantified boolean formula is true iff a particular sentence F is contained in some admissible argument of T.

The language of T contains atoms $p_1,\ldots,p_n$, $q_1,\ldots,q_m$, and $r_1,\ldots,r_k$ as well as atoms $t_1,\ldots,t_n$, intuitively holding if a truth value for the variables $p_1,\ldots,p_n$, has been chosen. Finally, there is an atom s that is used to inhibit that any admissible argument can choose truth values for the q_j. T consists of the following sentences:

$$\begin{aligned}
\neg L\neg p_i &\rightarrow p_i \wedge t_i,\\
\neg L p_i &\rightarrow \neg p_i \wedge t_i,\\
\neg L\neg \Phi,& \\
\neg L\neg q_j &\rightarrow q_j \wedge s \wedge \neg L s,\\
\neg L q_j &\rightarrow \neg q_j \wedge s \wedge \neg L s,
\end{aligned}$$

for each $i = 1,\ldots,n$, $j = 1,\ldots,m$.

We prove that there exists an admissible extension containing $F = \bigwedge t_i$ iff the 3-QBF formula is true.

Let us consider the case that the above T has an admissible extension $Th(T \cup \Delta)$ containing $\bigwedge t_i$. This implies that for each p_i either $\neg L\neg p_i$ or $\neg L p_i$ is part of the assumption set Δ. Further $\neg L\neg\Phi$ must be part of Δ in order for Δ to be closed. None of the assumptions $\neg L\neg q_i$ or $\neg L q_i$ can be part of Δ, however, as

otherwise Δ would attack itself.

Let $\Delta^\star$ be an assumption set attacking the admissible Δ above, and assume that $\Delta^\star$ attacks $\neg L\neg\Phi$ in Δ. Then $\Delta^\star$ together with the theory T would allow for the derivation of $\neg\Phi$. If $\Delta^\star$ makes choices for the p_i in conflict with the ones made by Δ, then Δ automatically attacks $\Delta^\star$. If $\Delta^\star$ makes the same choices for the p_i as Δ, then $\Delta^\star$ must include assumptions from the set $\{\neg L\neg q_j, \neg Lq_j\}_{j=1,\ldots,m}$. However, no such assumption can be counter-attacked, as otherwise Δ would be self-attacking and thus non-admissible. Therefore, for the given choices on the p_i's in Δ, no choices for the q_j's exist that make $\neg\Phi$ true. In other words, for the given choice of the p_i's in Δ, and for all choices of the truth values for the q_j's, there exists an assignment of truth values to the r_l's that make Φ true, which implies that the 3-QBF formula must be true.

Conversely, if there is no admissible extension containing $\bigwedge t_i$, regardless of our choices for the p_i's, there is always an attack on $\neg L\neg\Phi$, which by the arguments presented above implies that the 3-QBF formula cannot be true.

Hence, this is a log-space reduction from 3-QBF to credulous reasoning under the admissibility semantics in AEL, which proves the first claim in the theorem.

In order to prove the second claim, we consider the theory $T' = T \cup \{L\bigwedge_i t_i\}$, where T is the theory constructed from Φ. Any admissible set Δ must contain the assumptions $\neg L\neg\Phi$ and $L\bigwedge_i t_i$ in order for Δ to be closed. Furthermore, any admissible extension of T' must contain $\bigwedge_i t_i$ because otherwise it is attacked by $\neg L\bigwedge_i t_i$ without having a counter-attack. From this fact and the above observations it follows that T' has an admissible extension iff the 3-QBF formula is true. Given that if no admissible extension exists all co-sceptical queries will be answered negatively, the above is equivalent to the fact that $\neg\bigwedge_i t_i$ is not a sceptical consequence of T' iff the 3-QBF formula is true, i.e., the construction is a log-space reduction from 3-QBF to co-sceptical reasoning under the admissibility semantics. From that and the upper bound in Theorem 5, the second claim follows. ■

Applying Proposition 3, we get the following corollary.

Corollary 11 *Credulous reasoning in AEL under the preferability semantics is Σ_3^p-complete.*

Sceptical reasoning under the preferability semantics is even harder, since for each candidate preferred argument we have to verify that none of its supersets is admissible, resulting in a reasoning problem on the fourth level of the polynomial hierarchy.

Theorem 12 *Sceptical reasoning in AEL under the preferability semantics is Π_4^p-complete.*

Proof: Membership follows from Theorem 5.

Hardness will be proven by reducing 4-QBF to co-sceptical reasoning. Assume the following quantified boolean formula: $\exists p_1,\ldots,p_n\ \forall q_1,\ldots,q_m\ \exists r_1,\ldots,r_k\ \forall s_1,\ldots,s_o\ \Phi$, with Φ a formula in 3CNF over the propositional variables $p_1,\ldots,p_n$, $q_1,\ldots,q_m$, $r_1,\ldots,r_k$, and $s_1,\ldots,s_l$. We construct an AEL theory T such that the given quantified boolean formula is true iff a particular sentence F is not contained in some preferred argument of T.

The language of T contains atoms $p_1,\ldots,p_n$, $q_1,\ldots,q_m$, $r_1,\ldots,r_k$ and $s_1,\ldots,s_o$ as well as atoms $t_1,\ldots,t_m$, the latter intuitively holding if a truth value for the variables $q_1,\ldots,q_m$ has been chosen. Finally, we have atoms v, used to block the truth assignment to the q_j, and w, used to prohibit any choices on assumptions $\{\neg L\neg r_h, \neg Lr_h\}$.

T consists of the following sentences:

$$\begin{aligned}
\neg L\neg p_i &\rightarrow p_i,\\
\neg Lp_i &\rightarrow \neg p_i,\\
\neg L\neg q_j \wedge \neg Lv &\rightarrow q_j \wedge t_j,\\
\neg Lq_j \wedge \neg Lv &\rightarrow \neg q_j \wedge t_j,\\
\neg Lt_j &\rightarrow v,\\
\neg L\neg r_h &\rightarrow r_h \wedge w \wedge \neg Lw,\\
\neg Lr_h &\rightarrow \neg r_h \wedge w \wedge \neg Lw,\\
\Phi &\rightarrow v,
\end{aligned}$$

for each $i = 1,\ldots,n$, $j = 1,\ldots,m$, $h = 1,\ldots,k$.

We prove that there exists a preferred extension not containing $F = \bigwedge t_i$ iff the 4-QBF formula is true.

Let us construct one such preferred extension. First, note that any assumption set containing non-conflicting assumptions from the set $\{\neg L\neg p_i, \neg Lp_i\}$ is an admissible set. Let Δ be a maximal such set.

Secondly, the set Δ cannot be expanded using assumptions from $\{\neg L\neg r_h, \neg Lr_h\}$, because these assumptions lead to an immediate self-attack.

Thirdly, it is obvious that Δ can be expanded (in a non-trivial way) only by adding the assumption $\neg Lv$ and assumptions from the set $\{\neg L\neg q_j, \neg Lq_j\}$. Let us call this expanded set Δ'. Such a set Δ' is only admissible if we make choices for all q_j's because otherwise Δ' can be attacked by some $\{\neg Lt_j\}$ (using $\neg Lt_j \rightarrow v$) and Δ' cannot counter-attack such an attack. More-

over, such a set Δ' is only admissible if every attack against $\neg Lv$ can be counter-attacked by Δ'. The only attack $\Delta^\star$ against $\neg Lv$ which cannot be counter-attacked by Δ' would consists of all assumptions in Δ' and assumptions from $\{\neg L\neg r_h, \neg Lr_h\}$ such that Φ is true. Thus, assuming that Δ' is admissible means that no such $\Delta^\star$ exists, i.e. for all possible choices for the r_h's, Φ is not derivable, i.e., there is always a truth assignment to the s_k's that make $\neg\Phi$ true. This means that Δ cannot be expanded to Δ' by assumptions from $\{\neg L\neg q_j, \neg Lq_j\}$ and by $\neg Lv$ if, under the truth assignment to the p_i's corresponding to the assumptions in Δ, and for all truth assignments to the q_j's, there exists a truth assignment to the r_h's that makes Φ true. In other words, if there exists a preferred assumption set Δ' such that $Th(T \cup \Delta')$ does not contain $\bigwedge t_j$, then the given 4-QBF formula is true.

Conversely, let us assume the 4-QBF formula is true. Let Δ be an assumption set containing the assumptions from $\{\neg L\neg p_i, \neg Lp_i\}$ corresponding to a truth assignment to the p_i's rendering $\forall q_1,\ldots,q_m \exists r_1,\ldots,r_k \forall s_1,\ldots,s_o \Phi$ true. Δ is admissible. Moreover, any extension Δ' of Δ by assumptions from $\{\neg L\neg q_j, \neg Lq_j\}$ together with $\neg Lv$ would not be admissible. Indeed, for any such Δ' there exists a value assignment to the r_h's which makes Φ true, corresponding to an assumption set with choices from $\{\neg L\neg r_h, \neg Lr_h\}$ which, together with Δ', would attack Δ' without being counter-attacked by Δ'. For this reason, there exists a preferred extension not containing both $\neg Lv$ and choices from $\{\neg L\neg q_j, \neg Lq_j\}$, and hence this preferred extension does not contain $\bigwedge t_j$.

Hence, this is a log-space reduction from 4-QBF to co-sceptical reasoning under the preferability semantics in AEL, which proves the claim. ■

6 Conclusion and Discussion

We have analysed the computational complexity of credulous and sceptical reasoning under the admissibility and preferability semantics for (the propositional variant of) the nonmonotonic frameworks of Theorist, Circumscription and Autoepistemic Logic. Table 1 summarises the results.

Table 1: Complexity results

Framework	Property	Admissibility		Preferability	
		cred.	scept.	cred.	scept.
Theorist	simple &	Σ_2^p	co-NP	Σ_2^p	Π_2^p
Circum.	normal	Σ_2^p	co-NP	Σ_2^p	Π_2^p
AEL	general	Σ_3^p	Π_3^p	Σ_3^p	Π_4^p

These results imply that, for the *simple* and *normal* frameworks of Theorist and Circumscription, credulous reasoning is as hard under the new semantics as it is under the standard, stability semantics [2], whereas sceptical reasoning is simpler under the new semantics than it is under the standard semantics, but it is a trivial form of nonmonotonic reasoning as it amounts to reasoning in classical logic, the monotonic logic underlying both Theorist and Circumscription. Moreover, for the "general" framework of Autoepistemic logic, reasoning under the new semantics is considerably harder than under the standard semantics, as credulous reasoning under both new semantics is located *one level higher* in the polynomial hierarchy than credulous reasoning under the standard semantics, and sceptical reasoning under the admissibility and preferability semantics is located *one and two level higher* (respectively) in the polynomial hierarchy than sceptical reasoning under the standard semantics.

These results suggest that neither of the new semantics is appropriate for reasoning in general frameworks, that the admissibility semantics is not appropriate for sceptical reasoning in simple and normal frameworks, and that, in all other cases, no gain is obtained by reasoning under the new semantics with respect to reasoning under the standard semantics. These conclusions confirm those drawn from earlier results for logic programming and default logic in [Eiter *et al.*, 1998; Dimopoulos *et al.*, 1999].

These results seem to contradict the expectation, put forwards in [Bondarenko *et al.*, 1997; Dung *et al.*, 1997], that the new, argumentation-theoretic semantics lead to computationally less involved nonmonotonic reasoning processes than the standard semantics. However, they do not contradict the expectation that in practice credulous reasoning under the admissibility semantics is often easier than credulous reasoning under the stability semantics, especially if the frameworks employed are *flat* (e.g. logic programming and default logic). Indeed, [Kowalski and Toni, 1996] envisages practical and legal reasoning applications for the argumentation-theoretic semantics, where credulous reasoning under the admissibility semantics in flat frameworks is well-suited, as unilateral arguments are put forwards and defended against all counterarguments, in a credulous manner.

Future work include identifying, by means of empirical investigations, the practical impact of the complexity results proven in this paper as well as in [Dimopoulos *et al.*, 1999] for the envisaged applications of the

[2] See Section 3 for a summary of the results for the standard semantics.

semantics here investigated.

Acknowledgements

The second author has been partially supported by the University of Cyprus, and the third author has been supported by the UK EPSRC project "Logic-Based Multi-Agent Systems" and by the EU KIT project "Computational Logic for Flexible Solutions to Applications".

References

[Bondarenko *et al.*, 1997] Andrei Bondarenko, Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. An abstract, argumentation-theoretic framework for default reasoning. *Artificial Intelligence*, 93(1–2):63–101, 1997.

[Cadoli and Schaerf, 1993] Marco Cadoli and Marco Schaerf. A survey of complexity results for non-monotonic logics. *The Journal of Logic Programming*, 17:127–160, 1993.

[Dimopoulos *et al.*, 1999] Yannis Dimopoulos, Bernhard Nebel, and Francesca Toni. Preferred arguments are harder to compute than stable extensions. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, Stockholm, Sweden, August 1999. Morgan Kaufmann.

[Dung *et al.*, 1997] Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. Argumentation-theoretic proof procedures for default reasoning. Technical report, Imperial College, London, UK, 1997.

[Dung, 1991] Phan Minh Dung. Negation as hypothesis: an abductive foundation for logic programming. In K. Furukawa, editor, *Proceedings of the 8th International Conference on Logic programming*, pages 3–17, Paris, France, 1991. MIT Press.

[Eiter *et al.*, 1998] Thomas Eiter, Nicola Leone, and Domenico Saccà. Expressive power and complexity of partial models for disjunctive deductive databases. *Theoretical Computer Science*, 206:181–218, 1998.

[Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and Intractability—A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, 1979.

[Gottlob, 1992] Georg Gottlob. Complexity results for nonmonotonic logics. *Journal for Logic and Computation*, 2(3):397–425, 1992.

[Kowalski and Toni, 1996] Robert A. Kowalski and Francesca Toni. Abstract argumentation. *Journal of Artificial Intelligence and Law*, 4(3–4):275–296, 1996.

[Marek and Truszczynski, 1993] Victor W. Marek and Miroslaw Truszczynski. *Nonmonotonic logic: context-dependent reasoning*. Springer-Verlag, Berlin, Heidelberg, New York, 1993.

[McCarthy, 1980] John McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1–2):27–39, 1980.

[Moore, 1985] Robert C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25:75–94, 1985.

[Niemelä, 1990] Ilkka Niemelä. Towards automatic autoepistemic reasoning. In *Logics in AI*, volume 478 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, Heidelberg, New York, 1990.

[Papadimitriou, 1994] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.

[Poole, 1988] David Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47, 1988.

[Saccà and Zaniolo, 1990] Domenico Saccà and Carlo Zaniolo. Stable models and non-determinism in logic programs with negation. In *Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database-Systems (PODS-90)*, pages 205–217, 1990.

Complexity Results for Default Reasoning from Conditional Knowledge Bases

Thomas Eiter and **Thomas Lukasiewicz**
Institut und Ludwig Wittgenstein Labor für Informationssysteme, TU Wien
Favoritenstraße 9-11, A-1040 Vienna, Austria
{eiter, lukasiewicz}@kr.tuwien.ac.at

Abstract

Conditional knowledge bases have been proposed as belief bases that include defeasible rules (also called defaults) of the form "$\phi \rightarrow \psi$", which informally read as "generally, if ϕ then ψ." Such rules may have exceptions, which can be handled in different ways. A number of entailment semantics for conditional knowledge bases have been proposed in the literature. However, while the semantic properties and interrelationships of these formalisms are quite well understood, about their algorithmic properties only partial results are known so far. In this paper, we fill these gaps and draw a precise picture of the complexity of default reasoning from conditional knowledge bases: Given a conditional knowledge base KB and a default $\phi \rightarrow \psi$, does KB entail $\phi \rightarrow \psi$? We classify the complexity of this problem for a number of well-known approaches (including Goldszmidt et al.'s maximum entropy approach and Geffner's conditional entailment). We consider the general propositional case as well as syntactic restrictions (in particular, to Horn and literal-Horn conditional knowledge bases). Furthermore, we analyze the effect of precomputing rankings for the respective approaches. Our results complement and extend previous results, and contribute in exploring the tractability/intractability frontier of default reasoning from conditional knowledge bases.

1 INTRODUCTION

A conditional knowledge base consists of a collection of strict statements in classical logic and a collection of defeasible rules (also called defaults). The former are statements that must always hold, while the latter are rules $\phi \rightarrow \psi$ that read as "generally, if ϕ then ψ." Such rules may have exceptions, which can be handled in different ways. For example, the knowledge "penguins are birds" and "penguins don't fly" can be represented by strict sentences, while the knowledge "birds fly" should be expressed by a defeasible rule (since penguins are birds that do not fly).

The semantics of a conditional knowledge base KB is given by the set of all defaults that are plausible consequences of KB. The literature contains several different proposals for plausible consequence relations and extensive work on their desired properties. The core of these properties are the rationality postulates proposed by Kraus, Lehmann, and Magidor [34].

It turned out that these rationality postulates constitute a sound and complete axiom system for several classical model-theoretic entailment relations under uncertainty measures on worlds: They characterize classical model-theoretic entailment under preferential structures [45, 34], infinitesimal probabilities [1, 42], possibility measures [14], and world rankings [46, 27]. Moreover, they characterize an entailment relation based on conditional objects [15]. A survey of all these relationships is given in [5]. We will use the notion of ε-entailment to refer to these equivalent entailment relations. That their equivalence is not incidental is shown by Friedman and Halpern [18], who prove that many approaches are expressible as plausibility measures and thus they must, under some weak natural conditions, inevitably amount to the same notion of inference.

Mainly to solve problems with irrelevant information, the notion of rational closure as a more adventurous notion of entailment has been introduced by Lehmann [37, 39]. It is equivalent to entailment in system Z by Pearl [43] (which is generalized to variable strength defaults in system Z^+ by Goldszmidt and Pearl [26, 28]), to the least specific possibility entailment by Benferhat et al. [4], and to a conditional (modal) logic-based entailment by Lamarre [36].

Finally, mainly in order to solve problems with property inheritance from classes to exceptional subclasses, the max-

imum entropy approach to default entailment was proposed by Goldszmidt et al. [24] (and recently generalized to variable strength defaults by Bourne and Parsons [7]); the notion of lexicographic entailment was introduced by Lehmann [38] and Benferhat et al. [3]; the notion of conditional entailment was proposed by Geffner [20, 22]; and, an infinitesimal belief function approach was suggested by Benferhat et al. [6].

While the semantic properties and interrelationships of the various formalisms are quite well understood, their algorithmic properties are less explored. Algorithms for conditional knowledge bases have been described, for example, in [25, 39, 28, 12]. They are often used for a rough analysis of the computational complexity of the problems they solve. This way, in many cases, only rough upper bounds for the complexity of various computational problems have been established so far.

The aim of this paper is to fill these gaps and to draw a precise picture of the computational complexity of major formalisms for default reasoning from conditional knowledge bases. It thus complements and extends the previous work in [25, 39, 28, 12].

Our effort serves several purposes. Firstly, precise computational relationships between various formalisms are established, that is, the feasibility of a polynomial time transformation of reasoning in one formalism into reasoning in another one can be assessed from our complexity results. Secondly, the results show that certain algorithms in the literature have optimal order under worst case complexity. Finally, the results are useful when constructing new algorithms for default reasoning.

The main contributions of this paper are as follows:

- We give a sharp characterization of the complexity of default reasoning from conditional knowledge bases under several semantics, improving on previous results. In particular, we address the following generic problem: Given a conditional knowledge base KB and a default $\phi \rightarrow \psi$, is it true that KB entails $\phi \rightarrow \psi$? Note that the precise formulation of this problem slightly varies in the different approaches and may involve further parameters.

- Our analysis includes formalisms for which only very rough or even no complexity results have been derived so far, namely proper ϵ-entailment [25], maximum entropy entailment [24] together with its variable-strength extension [7], and Geffner's conditional entailment [20].

- We reconsider Lehmann's lexicographic entailment [38], and correct an error of [12] in the derivation of the complexity of generalized lexicographic entailment [3].

- We analyze the effect of compilation for ranking-based approaches, in terms of off-line computation of the ranking implicitly associated with the defaults in the knowledge base, such that it can be used on-line for default reasoning. Both the cost of computing the ranking and of its on-line use for default reasoning are examined.

- We analyze the impact of syntactical restrictions on the knowledge bases. In particular, we consider the restriction to the Horn case, where all strict statements are Horn clauses and all defeasible rules are of the form $\phi \rightarrow \psi$ with conjunctions of atoms ϕ and conjunctions of Horn clauses ψ, and the restriction to the literal-Horn case, where ψ is additionally a literal.

Our main findings can be briefly summarized as follows.

- The approaches considered in this paper cover different complexity classes at the low end of the polynomial hierarchy, which range from co-NP (ϵ-entailment) to Π_2^P (Geffner's conditional entailment). In general, they have lower complexity than well-known formalisms of nonmonotonic reasoning such as default logic, circumscription, or autoepistemic logic [48, 29, 16].

- As for the worst case, off-line computation of rankings does in general not pay off w.r.t. worst case complexity, and in particular does not buy tractability. Furthermore, computing the ranking associated with a knowledge base is as difficult as solving the reasoning problem.

- Horn constraints have different effects on the various semantics. For some approaches, the restriction to the Horn case leads to tractability, while for the others, the complexity remains unchanged. Interestingly, for all semantics, Horn and literal-Horn knowledge bases have the same complexity. In particular, Geffner's conditional entailment is Π_2^P-complete in the literal-Horn case, and thus harder than Reiter's default logic in this case [33, 47].

2 PRELIMINARIES

2.1 CONDITIONAL KNOWLEDGE BASES

We assume a set of *atoms* $At = \{p_1, \ldots, p_n\}$ with $n \geq 1$. By $\bot$ and $\top$ we denote the propositional constants *false* and *true*, respectively. The set of *classical formulas* is the closure of $At \cup \{\bot, \top\}$ under the Boolean operations $\neg$ and $\wedge$. We use $(\phi \Rightarrow \psi)$ and $(\phi \vee \psi)$ to abbreviate $\neg(\phi \wedge \neg\psi)$ and $\neg(\neg\phi \wedge \neg\psi)$, respectively, and adopt the usual conventions to eliminate parentheses. A *literal* is an atom p from At or its negation $\neg p$. A *Horn clause* is a classical formula $\phi \Rightarrow \psi$, where ϕ is either $\top$ or a conjunction of atoms, and ψ is either $\bot$ or an atom.

A *conditional rule* (or *default*) is an expression $\phi \rightarrow \psi$, where ϕ and ψ are classical formulas. A *conditional knowledge base* is a pair $KB = (L, D)$, where L is a finite set of classical formulas and D is a finite set of defaults. Informally, L contains facts and rules that are certain, while

D contains defeasible rules. In case $L = \emptyset$, we call KB a *default knowledge base*. A default $\phi \rightarrow \psi$ is *Horn* (resp., *literal-Horn*), if ϕ is either $\top$ or a conjunction of atoms, and ψ is a conjunction of Horn clauses (resp., ψ is a literal).

Given a conditional knowledge base $KB = (L, D)$, a *strength assignment* σ on KB is a mapping that assigns each default $d \in D$ a nonnegative integer $\sigma(d)$. A *priority assignment* on KB is a strength assignment π on KB such that $\{\pi(d) \mid d \in D\} = \{0, 1, \ldots, k\}$ for some $k \geq 0$. Informally, a priority assignment is a strength assignment in which there are no "empty levels".

An *interpretation* (or *world*) is a truth assignment I: $At \rightarrow \{\textbf{true}, \textbf{false}\}$, which is extended to classical formulas as usual. We use $\mathcal{I}_{At}$ to denote the set of all worlds for At. The world I *satisfies* a classical formula ϕ, or I is a *model* of ϕ, denoted $I \models \phi$, iff $I(\phi) = \textbf{true}$. I *satisfies* a default $\phi \rightarrow \psi$, or I is a *model* of $\phi \rightarrow \psi$, denoted $I \models \phi \rightarrow \psi$, iff $I \models \phi \Rightarrow \psi$. I *satisfies* a set K of classical formulas and defaults, or I is a *model* of K, denoted $I \models K$, iff I satisfies every formula in K. The world I *verifies* a default $\phi \rightarrow \psi$, denoted $I \models_{\mathrm{v}} \phi \rightarrow \psi$, iff $I \models \phi \wedge \psi$. I *falsifies* a default $\phi \rightarrow \psi$, iff $I \models \phi \wedge \neg\psi$ (that is, $I \not\models \phi \rightarrow \psi$). A set of defaults D *tolerates* a default d *under* a set of classical formulas L iff $D \cup L$ has a model that verifies d. A set of defaults D is *under* L *in conflict* with a default $\phi \rightarrow \psi$ iff all models of $D \cup L \cup \{\phi\}$ satisfy $\neg\psi$.

A *world ranking* κ is a mapping κ: $\mathcal{I}_{At} \rightarrow \{0, 1, \ldots\} \cup \{\infty\}$ such that $\kappa(I) = 0$ for at least one world I. It is extended to all classical formulas ϕ as follows. If ϕ is satisfiable, then $\kappa(\phi) = \min\{\kappa(I) \mid I \in \mathcal{I}_{At}, I \models \phi\}$; otherwise, $\kappa(\phi) = \infty$. A world ranking κ is *admissible* with a conditional knowledge base (L, D) iff $\kappa(\neg\phi) = \infty$ for all $\phi \in L$, and $\kappa(\phi) < \infty$ and $\kappa(\phi \wedge \psi) < \kappa(\phi \wedge \neg\psi)$ for all defaults $\phi \rightarrow \psi \in D$. A *default ranking* σ on D maps each $d \in D$ to an integer.

2.2 SEMANTICS FOR CONDITIONAL KNOWLEDGE BASES

We now recall some semantics for conditional knowledge bases. To simplify the presentation, we shall adjust original definitions (without significant effects) to our framework.

2.2.1 ε-Semantics (Adams [1] and Pearl [42])

We describe the notions of ε-consistency, ε-entailment, and proper ε-entailment in terms of world rankings (see especially Geffner's work [20, 21] for the equivalence to the original definitions).

A conditional knowledge base KB is *ε-consistent* iff there exists a world ranking that is admissible with KB. It is *ε-inconsistent* iff no such a world ranking exists.

A conditional knowledge base KB *ε-entails* a default $\phi \rightarrow \psi$ iff either $\kappa(\phi) = \infty$ or $\kappa(\phi \wedge \psi) < \kappa(\phi \wedge \neg\psi)$ for all world rankings κ that are admissible with KB. Moreover, KB *properly ε-entails* the default $\phi \rightarrow \psi$ iff KB ε-entails $\phi \rightarrow \psi$ and KB does not ε-entail $\phi \rightarrow \bot$.

The next theorem is a simple generalization of a result by Adams [1], who stated it for $L = \emptyset$.

Theorem 2.1 (essentially [1]) *A conditional knowledge base (L, D) ε-entails a default $\phi \rightarrow \psi$ iff the conditional knowledge base $(L, D \cup \{\phi \rightarrow \neg\psi\})$ is ε-inconsistent.*

2.2.2 Systems Z and Z^+ (Pearl [43] and Goldszmidt and Pearl [26, 28])

The notions of entailment in systems Z and Z^+ apply to ε-consistent conditional knowledge bases $KB = (L, D)$. Entailment in system Z^+ additionally assumes a strength assignment σ on KB. It is associated with a default ranking z^+ and a world ranking κ^+, which are defined as the unique solution of the following system of equations: For all $d = \phi \rightarrow \psi \in D$ and all $I \in \mathcal{I}_{At}$:

$$z^+(d) = \sigma(d) + \kappa^+(\phi \wedge \psi) \quad (1)$$

$$\kappa^+(I) = \begin{cases} \infty & \text{if } I \not\models L \\ 0 & \text{if } I \models L \cup D \\ 1 + \max\limits_{d \in D:\, I \not\models d} z^+(d) & \text{otherwise.} \end{cases} \quad (2)$$

A default $\phi \rightarrow \psi$ is *z^+-entailed* by (KB, σ) *at strength* τ iff either $\kappa^+(\phi) = \infty$ or $\kappa^+(\phi \wedge \psi) + \tau < \kappa^+(\phi \wedge \neg\psi)$.

Finally, z-entailment is a special case of z^+-entailment: A default $\phi \rightarrow \psi$ is *z-entailed* by KB iff $\phi \rightarrow \psi$ is z^+-entailed by (KB, σ) at strength 0, where σ is given by $\sigma(d) = 0$ for all $d \in D$. In this special case, the rankings z^+ and κ^+ are denoted by z and κ^z, respectively.

2.2.3 Maximum Entropy Semantics (Goldszmidt et al. [24] and Bourne and Parsons [7])

The notion of z^*-entailment applies to ε-consistent minimal-core conditional knowledge bases $KB = (L, D)$ without strength assignment, where KB is *minimal-core* iff for each default $d \in D$ there exists a model I of $L \cup (D - \{d\})$ that falsifies d. This notion of entailment is linked to a default ranking z^* and a world ranking κ^*, which are defined as the unique solution of the following system of equations: For all $d = \phi \rightarrow \psi \in D$ and all $I \in \mathcal{I}_{At}$:

$$z^*(d) = 1 + \kappa^*(\phi \wedge \psi) \quad (3)$$

$$\kappa^*(I) = \begin{cases} \infty & \text{if } I \not\models L \\ 0 & \text{if } I \models L \cup D \\ \sum\limits_{d \in D:\, I \not\models d} z^*(d) & \text{otherwise.} \end{cases} \quad (4)$$

A default $\phi \rightarrow \psi$ is *z^*-entailed* by KB iff either $\kappa^*(\phi) = \infty$ or $\kappa^*(\phi \wedge \psi) < \kappa^*(\phi \wedge \neg\psi)$.

The notion of z_s^*-entailment applies to ϵ-consistent conditional knowledge bases $KB = (L, D)$ with additional positive strength assignment σ. It is defined whenever the following system of equations (5) and (6) has a unique solution z_s^* and κ_s^*. For all $\phi \to \psi \in D$ and all $I \in \mathcal{I}_{At}$:

$$\kappa_s^*(\phi \wedge \neg\psi) = \sigma(\phi \to \psi) + \kappa_s^*(\phi \wedge \psi) \quad (5)$$

$$\kappa_s^*(I) = \begin{cases} \infty & \text{if } I \not\models L \\ 0 & \text{if } I \models L \cup D \\ \sum_{d \in D:\, I \not\models d} z_s^*(d) & \text{otherwise.} \end{cases} \quad (6)$$

The uniqueness of z_s^* and κ_s^* is guaranteed by assuming that κ_s^* is *robust* [7], which is the following property: for all distinct defaults d_1 and d_2 in D, it holds that all models I_1 and I_2 of L having smallest ranks in κ_s^* such that $I_1 \not\models d_1$ and $I_2 \not\models d_2$, respectively, are different. That is, d_1 and d_2 do not have a common minimal falsifying model under L.

A default $\phi \to \psi$ is *z_s^*-entailed* by (KB, σ) *at strength* τ iff either $\kappa_s^*(\phi) = \infty$ or $\kappa_s^*(\phi \wedge \psi) + \tau \leq \kappa_s^*(\phi \wedge \neg\psi)$.

The notion of z_s^*-entailment is a proper generalization of z^*-entailment:

Lemma 2.2 *Let $KB = (L, D)$ with strength assignment $\sigma(d) = 1$ for all $d \in D$ be ϵ-consistent and minimal-core. Then the system of equations given by (5) and (6) for all $\phi \to \psi \in D$ and all $I \in \mathcal{I}_{At}$ has a unique solution z_s^*, κ_s^*, which coincides with z^*, κ^*. Moreover, κ_s^* is robust.*

2.2.4 Lexicographic Entailment (Lehmann [38] and Benferhat et al. [3])

Lexicographic entailment as introduced in [3] applies to conditional knowledge bases $KB = (L, D)$ with a priority assignment π on KB, which defines an ordered partition $(D_0, \ldots, D_k)$ of D by $D_i = \{d \in D \mid \pi(d) = i\}$, for all $i \leq k$. It is used to define a *preference ordering* on worlds as follows. A world I is *π-preferable* to a world I' iff there exists some $i \in \{0, \ldots, k\}$ such that $|\{d \in D_i \mid I \models d\}| > |\{d \in D_i \mid I' \models d\}|$ and $|\{d \in D_j \mid I \models d\}| = |\{d \in D_j \mid I' \models d\}|$ for all $i < j \leq k$. Note that this preference ordering can be expressed by a world ranking. A model I of a set of classical formulas $\mathcal{F}$ is a *π-preferred* model of $\mathcal{F}$ iff no model of $\mathcal{F}$ is π-preferable to I.

A default $\phi \to \psi$ is *lex_p-entailed* by (KB, π) iff ψ is satisfied in every π-preferred model of $L \cup \{\phi\}$. We will omit π when it is clear from the context.

The notion of lexicographic entailment in [38] is a special case of lexicographic entailment as above. It uses a particular priority assignment that is logically entrenched in KB, namely the default ranking z of KB (see Section 2.2.2). We then say that a default $\phi \to \psi$ is *lex-entailed* by KB iff $\phi \to \psi$ is *lex_p-entailed* by (KB, z). Note that this definition assumes that KB is ϵ-consistent.

It appears that, in a certain sense, lex-entailment is not less expressive than lex_p-entailment. That is, under a weak condition, priority assignments are expressible through logical entrenchment:

Theorem 2.3 *Let $KB = (L, D)$ such that every $d \in D$ has a verifying world, and let π be a priority assignment on KB. Then, some $KB' = (L', D')$ and ϕ' exist such that, for any default $\phi \to \psi$ over At, it holds that (KB, π) lex_p-entails $\phi \to \psi$ iff KB' lex-entails $\phi \wedge \phi' \to \psi$.*

2.2.5 Conditional Entailment (Geffner [20, 22])

Given a conditional knowledge base $KB = (L, D)$, a *priority ordering* $\prec$ on D is an irreflexive and transitive binary relation on D. We say $\prec$ is *admissible* with KB iff each set of defaults $D' \subseteq D$ that is under L in conflict with some default $d \in D$ contains a default d' such that $d' \prec d$.

Based on $\prec$, we define a *preference ordering* on worlds as follows. A world I is *$\prec$-preferable* to a world I', denoted $I \prec I'$, iff $\{d \in D \mid I \not\models d\} \neq \{d \in D \mid I' \not\models d\}$ and for each default $d \in D$ such that $I \not\models d$ and $I' \models d$, there exists a default $d' \in D$ such that $d \prec d'$, $I \models d'$, and $I' \not\models d'$. A model I of a set of classical formulas $\mathcal{F}$ is a *$\prec$-preferred* model of $\mathcal{F}$ iff no model of $\mathcal{F}$ is $\prec$-preferable to I.

A default $\phi \to \psi$ is *conditionally entailed* by KB iff ψ is satisfied in every $\prec$-preferred model of $L \cup \{\phi\}$ of every priority ordering $\prec$ that is admissible with KB.

2.3 COMPLEXITY CLASSES

We assume some familiarity with P, NP, and co-NP. We now briefly introduce some other classes that we encounter in our analysis (see [41, 32, 44, 31] for further background).

The class $\text{P}^{\text{NP}} = \Delta_2^\text{P}$ (resp., $\text{NP}^{\text{NP}} = \Sigma_2^\text{P}$) contains all decision problems that can be solved in deterministic (resp., nondeterministic) polynomial time with an oracle for NP. The class Π_2^P is the complementary class of Σ_2^P, which has Yes- and No-instances interchanged. The class D^P contains the problems that can be described as a logical conjunction of a problem in NP and a problem in co-NP. Any problem in D^P can be solved with two NP oracle calls, and is intuitively easier than a P^{NP}-complete problem. The class $\Delta_2^\text{P}[O(\log n)]$ contains the problems in P^{NP} that can be solved with $O(\log n)$ many oracle calls, where n is the size of the problem input. This class coincides with $\text{P}_{\|}^{\text{NP}}$, that is, polynomial time computability with parallel NP oracle calls. According to the current belief in complexity theory, the following is a strict hierarchy of inclusions:

$$\text{P} \subseteq \text{NP}, \text{co-NP} \subseteq \text{D}^\text{P} \subseteq \Delta_2^\text{P}[O(\log n)] = \text{P}_{\|}^{\text{NP}} \subseteq \text{P}^{\text{NP}} \subseteq \Sigma_2^\text{P}, \Pi_2^\text{P}.$$

The classes FP, $\mathrm{FP}_{\|}^{\mathrm{NP}}$, and $\mathrm{FP}^{\mathrm{NP}}$ are the functional analogs of P, $\mathrm{P}_{\|}^{\mathrm{NP}}$, and P^{NP}, respectively.

Completeness for decision and function classes is w.r.t. standard polynomial time transformations (for the latter, see [35, 31]). In case of P and FP, completeness is w.r.t. logspace-reductions.

3 OVERVIEW OF COMPLEXITY RESULTS

3.1 PROBLEM STATEMENTS AND RESULTS

A *default reasoning problem* is a pair (KB, d), where $KB = (L, D)$ is a conditional knowledge base and d is a default. It is *Horn* iff L is a finite set of Horn clauses, D is a finite set of Horn defaults, and d is a Horn default. It is *literal-Horn* iff L is a finite set of Horn clauses, D is a finite set of literal-Horn defaults, and d is a literal-Horn default. In case of z^+- and z^*-entailment, we assume that KB and d have a strength assignment $\sigma(KB)$ and a strength $\tau(d)$, respectively. In case of lex_p-entailment, we assume that KB has a priority assignment $\pi(KB)$. We tacitly assume that KB satisfies any preconditions that the definition of entailment may request.

We analyze the complexity of the following problems:

- ENTAILMENT: Given a default reasoning problem (KB, d), we are asked whether KB entails d under some fixed semantics $\mathcal{S}$. In case of z^+- and z^*-entailment, we are asked whether d is z^+- and z^*-entailed, respectively, by $(KB, \sigma(KB))$ at strength $\tau(d)$. In case of lex_p-entailment, we are asked whether d is lex_p-entailed by $(KB, \pi(KB))$.
- RANKING: Given a conditional knowledge base KB, we are asked to compute the default ranking R of KB according to some fixed semantics $\mathcal{S}$ (that is, the rank of each default in D).
- RANK-ENTAILMENT: Same as entailment, but the (unique) default ranking R of KB according to some fixed semantics $\mathcal{S}$ is part of the problem input.

The problems RANKING and RANK-ENTAILMENT are relevant from a preprocessing perspective, in which the ranking R of a conditional knowledge base is computed in advance and then on-line available in the input for solving an entailment problem.

Our results, together with results from the literature, are compactly summarized in Tables 1–3.

It appears that a number of different complexity classes from P up to Π_2^{P}, the second level of the polynomial hierarchy, are covered. A first observation is that Geffner's conditional entailment has the highest complexity (Π_2^{P}, Table 1) of all the formalisms considered in this paper. It is thus in the same league as the major formalisms of nonmonotonic reasoning, such as Reiter's default logic, circumscription, and McDermott and Doyle's nonmonotonic logic [29, 48, 16], which are all Π_2^{P}-complete. All other approaches in Table 1 have (considerably) lower complexity.

3.2 PREVIOUS RESULTS

Lehmann and Magidor [39] proved that preferential entailment (and thus also ε-entailment) for default knowledge bases is co-NP-complete. Moreover, Lehmann and Magidor [39] and Goldszmidt and Pearl [28] showed that ε-entailment for Horn default knowledge bases is in P. Finally, Goldszmidt and Pearl [25] proved that proper ε-entailment for general conditional knowledge bases (resp., Horn conditional knowledge bases) is in P^{NP} (resp., P).

As shown in [28], ENTAILMENT, RANKING, and RANK-ENTAILMENT in system Z^+ are in P, FP, and P, respectively, for Horn default knowledge bases, and in P^{NP}, $\mathrm{FP}^{\mathrm{NP}}$, and $\mathrm{P}_{\|}^{\mathrm{NP}}$, respectively, for general default knowledge bases. A fortiori, these upper bounds also hold for system Z.

Not much work has been spent determining the complexity of z^*- and z_s^*-entailment. Goldszmidt et al. [24] suspect that the complexity of z^*-entailment is high, and briefly note that, referring to results on Horn optimization problems, [2], the problem should be NP-hard in the Horn case.

Cayrol et al. show in [12] that lex_p-entailment is P^{NP}-complete for default knowledge bases. Moreover, they state in [12] that lex_p-entailment is P^{NP}-hard for Horn default knowledge bases. However, the short proof sketch in [12] is inappropriate, since it mentions a reduction from a problem that is obviously in $\mathrm{P}_{\|}^{\mathrm{NP}}$; this would only establish $\mathrm{P}_{\|}^{\mathrm{NP}}$-hardness for the Horn case. P^{NP}-hardness for the Horn case — but not the literal-Horn case — can be concluded from proofs for related recent results on the complexity of lexicographic belief revision [40].

To our knowledge, there are no complexity results on Geffner's conditional entailment so far.

Note that under a number of semantics for conditional knowledge bases inherited from conditional modal logics, entailment is known to be co-NP-complete, cf. [8, 19].

3.3 DISCUSSION

3.3.1 General Case

At the low end of the complexity range, there are ε-entailment, which has the same complexity as classical logic, and proper ε-entailment, which has marginally higher complexity due to the additional ε-entailment requirement. At the high end, we have Geffner's conditional entailment. Its high complexity is intuitively explained by

Table 1: Complexity of Deciding Entailment

	general case	Horn case	literal-Horn case
ε-entailment	co-NP-complete $^{+}$	P-complete **	P-complete **
proper ε-entailment	D^{P}-complete	P-complete *	P-complete *
z-entailment	$\mathrm{P}_{\parallel}^{\mathrm{NP}}$-complete	P-complete ***	P-complete ***
z^{+}-entailment	P^{NP}-complete ***	P-complete ***	P-complete ***
z^{*}-entailment	P^{NP}-complete	P^{NP}-complete	P^{NP}-complete
z_{s}^{*}-entailment	P^{NP}-complete	P^{NP}-complete	P^{NP}-complete
lex-entailment	P^{NP}-complete	P^{NP}-complete	P^{NP}-complete
$\mathrm{lex}_{\mathrm{p}}$-entailment	P^{NP}-complete $^{++}$	P^{NP}-complete $^{++}$	P^{NP}-complete
conditional entailment	Π_2^{P}-complete	Π_2^{P}-complete	Π_2^{P}-complete

Table 2: Complexity of Computing Default Rankings

	general case	Horn case	literal-Horn case
z	$\mathrm{FP}_{\parallel}^{\mathrm{NP}}$-complete	FP-complete ***	FP-complete ***
z^{+}	$\mathrm{FP}^{\mathrm{NP}}$-complete ***	FP-complete ***	FP-complete ***
z^{*}	$\mathrm{FP}^{\mathrm{NP}}$-complete	$\mathrm{FP}^{\mathrm{NP}}$-complete	$\mathrm{FP}^{\mathrm{NP}}$-complete
z_{s}^{*}	$\mathrm{FP}^{\mathrm{NP}}$-complete	$\mathrm{FP}^{\mathrm{NP}}$-complete	$\mathrm{FP}^{\mathrm{NP}}$-complete

Table 3: Complexity of Deciding Entailment Given the Default Rankings

	general case	Horn case	literal-Horn case
z-entailment	$\mathrm{P}_{\parallel}^{\mathrm{NP}}$-complete ***	P-complete ***	P-complete ***
z^{+}-entailment	$\mathrm{P}_{\parallel}^{\mathrm{NP}}$-complete ***	P-complete ***	P-complete ***
z^{*}-entailment	P^{NP}-complete	P^{NP}-complete	P^{NP}-complete
z_{s}^{*}-entailment	P^{NP}-complete	P^{NP}-complete	P^{NP}-complete

* Membership shown in [25].
** Membership shown in [39, 28] for default knowledge bases.
*** Membership was shown in [28] for default knowledge bases.
$^{+}$ Shown in [39] for default knowledge bases. Note that deciding ε-consistency is NP-complete in the general case [17].
$^{++}$ Reported in [12] for default knowledge bases; the proof sketch for the Horn case in [12] shows merely $\mathrm{P}_{\parallel}^{\mathrm{NP}}$-hardness.

an inherent pattern similar to reasoning under circumscription: To disprove that $KB = (L, D)$ entails $\phi \rightarrow \psi$, a $\prec$-preferred model I of $L \cup \{\phi\}$ under some admissible priority ordering $\prec$ must be found such that ψ is false in I. As it turns out, such a guess can be verified in polynomial time with an NP oracle, where the oracle checks the $\prec$-preferredness of I (i.e., minimality under $\prec$). This is similar to circumscription, i.e., minimal model reasoning, where for disproving $CIRC(\phi) \models \psi$ the minimality of a guessed model M of ϕ in which ψ is false must be verified, which is a co-NP-complete problem [9].

Also for the ranking-based approaches in Table 1, the problem of verifying whether a model I of a formula ϕ is selected on the basis of $KB = (L, D)$ is (at least) co-NP-hard in general. However, there is a qualitative difference between them and Geffner's approach. Each world ranking r induces a *modular partial ordering* on the models of $L \cup \{\phi\}$, in which *any two distinct models I_1 and I_2 of $L \cup \{\phi\}$ are comparable by their ranks $r(I_1)$ and $r(I_2)$*. The models with the same rank form a cluster, and the clusters are totally ordered by these ranks. The "preferred" models I of $L \cup \{\phi\}$ are those which have minimal rank $r(I)$. Using an NP oracle, it is possible to compute this minimal rank $r(I)$ in polynomial time, which is a polynomial-size certificate for recognizing preferred models efficiently. Intuitively, we have here a single well-connected search space, in which all preferred models of $L \cup \{\phi\}$ can be nailed down by this certificate.

On the other hand, in Geffner's conditional entailment, two models $I_1 \neq I_2$ of $L \cup \{\phi\}$ may be incomparable, i.e., nei-

ther $I_1 \prec I_2$ nor $I_2 \prec I_1$ may hold. In general, the search space for a preferred model of $L \cup \{\phi\}$ generally splits into an *exponential number* of completely disconnected search spaces, each of which is intractable and may contain a preferred model we are looking for. Moreover, there is no certificate computable in polynomial time with an NP oracle such that we can recognize preferred models efficiently from it (unless the polynomial hierarchy collapses).

More precisely, Geffner's approach suffers from two sources of complexity: (i) the number of candidates for a preferred model I of $L \cup \{\phi\}$ in the possibly exponentially many disconnected search spaces, which are generated by incomparability of two models $I_1 \neq I_2$ of $L \cup \{\phi\}$ because of to default violation; and (ii) the possibly exponential number of models I' of $L \cup \{\phi\}$ that are preferred to I, according to the admissibility ordering $\prec$. Note that, different from expectation, classical inference is *not* listed as a principal source of complexity here, as results on the Horn restrictions (discussed below) show.

The mid-range of complexity is covered by the ranking-based approaches. Roughly speaking, for any classical formula α, the rank $r(\alpha)$ can be computed as follows:

1. Compute the default ranking R for KB;
2. Compute $r(\alpha) = \min_{I \models \alpha} r(I)$, using R.

Algorithms for computing R in Step 1 have been described in the literature. They can be reformulated to run in P^{NP} (see [17]). Step 2 is feasible in P^{NP} by doing binary search on the range of the possible values for $r(I)$. This means that the condition "$r(\phi) = \infty$ or $r(\phi \wedge \psi) < r(\phi \wedge \neg\psi)$" for the entailment of $\phi \rightarrow \psi$ from KB is decidable in P^{NP}, by simply checking the satisfiability of $L \cup \{\phi\}$, and if needed computing $r(\phi \wedge \psi)$, $r(\phi \wedge \neg\psi)$ and comparing them.

The complexity of Steps 1 and 2 is shown in Tables 2 and 3, respectively. As for Step 1, the rank $R(d)$ of a default d may range, except in case of z, over exponentially many possible values; in case of z, it ranges over $[0, \ldots, n-1]$ and thus over a linear number of values. Informally, the ranking R can be constructed bottom up, starting with defaults having lowest rank, and then computing the rank of the next default by doing a binary search on the range of its possible values. This resembles the $\mathrm{FP}^{\mathrm{NP}}$-complete problem of computing the lexicographic maximum model of a formula ϕ [35] and suggests that computing R has the same complexity. This is true in all cases except one. For z-entailment, exploiting the fact that the sum of all default ranks is minimal at z, this can be done in polynomial time using parallel NP queries (see [17]).

Table 3 tells us that in all cases except z^+, entailment does not become easier if the default ranking R is known. Thus, from a worst case perspective, precomputing the default ranking R does not pay off (but clearly saves time over repetitive computations). In case of z^+, entailment becomes easier, because only the order of the defaults in z^+ is relevant, but not their actual ranks (which can thus be replaced by values from $[0, \ldots, n-1]$, and thus z^+- reduces to z-entailment).

3.3.2 Horn and Literal-Horn Case

In all tables, the results for the Horn and the literal-Horn case are the same. Thus, although it is in general not possible to simply split a Horn default $\phi \rightarrow \psi_1 \wedge \cdots \wedge \psi_m$ into a semantically equivalent set of literal-Horn defaults $\phi \rightarrow \psi_1, \ldots, \phi \rightarrow \psi_m$, it is possible to rewrite a Horn default reasoning problem to a literal-Horn one in polynomial time. Thus, the restriction of the general Horn to the literal-Horn case does not decrease the complexity.

At the low end of the complexity range, the Horn restriction gives tractability, as was (essentially) shown in [25, 39, 28]. Whereas, at the high end, Geffner's conditional entailment has surprisingly its full complexity already in the literal-Horn case. This is exceptional, since related formalisms such as Reiter's default logic and circumscription have lower complexity (more precisely, co-NP) in the Horn case [33, 47, 11]. Informally, in Geffner's conditional entailment, the Horn property is not sufficient to eliminate the intractability of the preference check for a model of a formula, as this test is not simply reducible to a polynomial number of Horn satisfiability tests, as e.g. in Reiter's default logic. Further syntactic restrictions are needed. One such restriction, which is efficiently checkable, is that the admissibility ordering $\prec$ is void.

At the mid-range of complexity, the syntactic restrictions have different effects. Tractability for z- and z^+-entailment is gained since in the respective entailment algorithms, the NP oracle can be replaced by a polynomial time procedure. On the other hand, for z^*- and z^*_σ-entailment, the Horn restriction does not decrease complexity. The intuitive reason is that here the rank of a default is given by the smallest *sum of ranks* of a set of violated defaults plus a value, while in the former cases this was the *maximum rank* of a single violated default plus a value. Computing the smallest sum is an intractable optimization problem, as there are exponentially many such sets; as pointed out in [24], it makes z^* (thus also z^*_σ) entailment NP-hard in the Horn case. Contrary to this, the maximum violated rank can be computed by simply looping through the already ranked defaults.

4 DERIVATION OF RESULTS

Due to space reason, we focus here on maximum entropy semantics and conditional entailment. Detailed proofs of all results are given in the full paper [17].

Algorithm z_s^*-ranking (essentially [7])

Input: ε-consistent $KB = (L, D)$ with positive strength assignment σ.
Output: Default ranking z_s^*, if the system of equations (5) and (6), for all $\phi \rightarrow \psi \in D$ and $I \in \mathcal{I}_{At}$, has a unique solution z_s^*, κ_s^* such that κ_s^* is robust; otherwise, *nil*.
Notation: We use $\mathrm{minv}(\phi \rightarrow \psi)$ and $\mathrm{minf}(\phi \rightarrow \psi)$ to denote $\kappa_s^*(\phi \wedge \psi)$ and $\kappa_s^*(\phi \wedge \neg\psi)$, respectively, where $\kappa_s^*(\alpha) = \min_{I \in \mathcal{I}_{At}: I \models \alpha} \kappa_s^*(I)$ with $\kappa_s^*(I)$ as in (6).

1. **for each** $d \in D$ **do** $z_s^*(d) := \infty$;
2. **while** $\{d \in D \mid z_s^*(d) = \infty\} \neq \emptyset$ **do begin**
3. Take any $d \in D$ with $z_s^*(d) = \infty$ such that $\sigma(d) + \mathrm{minv}(d)$ is minimal;
4. $z_s^*(d) := 0$;
5. **if** $\mathrm{minf}(d) = \infty$ **then return** *nil*;
6. $z_s^*(d) := \sigma(d) + \mathrm{minv}(d) - \mathrm{minf}(d)$
7. **end**;
8. **if** κ_s^* satisfies (5) for all $\phi \rightarrow \psi \in D$ **and** κ_s^* is robust
9. **then return** z_s^* **else return** *nil*.

Figure 1: Algorithm z_s^*-ranking

4.1 MAXIMUM ENTROPY SEMANTICS

The key for the P^{NP}-upper bound of z_s^*-entailment (hence, also z^*-entailment) is algorithm $z^*{}_s$-**ranking** (see Fig. 1), adapted from [7]. It intializes all default ranks $z_s^*(d)$ to ∞ and then ranks one default at a time, using (6) and the current z_s^*. If the process fails, a special value *nil* is returned.

We show the following properties of algorithm z_s^*-**ranking**: (i) computing $\mathrm{minv}(d)$ and $\mathrm{minf}(d)$ in Steps 3, 5, and 6 is in $\mathrm{FP}^{\mathrm{NP}}$; (ii) deciding whether κ_s^* satisfies (5) in Step 8 is in P^{NP} for each $\phi \rightarrow \psi \in D$; and (iii) deciding whether κ_s^* is robust in Step 8 is in P^{NP}.

As for (i) and (ii), an inductive argument shows that $|z_s^*(d)| \leq s\, 2^{|D|}$, where $s = \max_{d \in D} \sigma(d)$, holds for each $d \in D$ that is assigned a value $< \infty$ (see Appendix). Thus, for every $\phi \rightarrow \psi \in D$, the values $\kappa_s^*(\phi \wedge \psi)$ and $\kappa_s^*(\phi \wedge \neg\psi)$ are from $\{0, \ldots, s \cdot 2^n\} \cup \{\infty\}$. By binary search, they are polynomially computable with $O(n + \log s)$ calls to an NP-oracle, since given w, deciding whether some $I \in \mathcal{I}_{At}$ satisfies $\kappa_s^*(I) \leq w$ is in NP. Thus, (i) and (ii) are in $\mathrm{FP}^{\mathrm{NP}}$ and P^{NP}, respectively. For (iii), we test whether there are no $d_1 \neq d_2 \in D$ that have a common minimal falsifying model under L. If $\mathrm{minf}(d)$ is known for all $d \in D$—which can be computed in $\mathrm{FP}^{\mathrm{NP}}$—this is clearly in co-NP, and thus (iii) is in P^{NP}. Overall, computing the z_s^*-ranking of KB is thus in $\mathrm{FP}^{\mathrm{NP}}$.

Now consider deciding whether KB z_s^*-entails a default $\phi \rightarrow \psi$, once z_s^* is known. Testing $\kappa_s^*(\phi) = \infty$ is in NP, and by similar arguments as above, computing $\kappa_s^*(\phi \wedge \psi)$ and $\kappa_s^*(\phi \wedge \neg\psi)$ in a binary search from z_s^* is in $\mathrm{FP}^{\mathrm{NP}}$. Finally, testing $\kappa_s^*(\phi \wedge \psi) < \kappa_s^*(\phi \wedge \neg\psi)$ is simple. Put together, deciding whether KB z_s^*-entails d is in P^{NP}.

For the matching lower bound, we reduce the following P^{NP}-complete problem to z^*-entailment. Given a set of weighted Horn clauses $C = \{\phi_i \mid 1 \leq i \leq m\}$ such that every ϕ_i is satisfiable and has weight $w_i = 2^{c_i}$, where $c_i \geq 0$ is an integer, and some $r \in \{1, \ldots, m\}$, decide whether $I \models \phi_r$ for every maximum weight world I for C, i.e., world $I \in \mathcal{I}_{At}$ such that $\sum_{I \models \phi_i} w_i$ is maximum. P^{NP}-hardness of this problem follows by a minor adaptation of the proof of Theorem 2.1 in [35].

We construct a literal-Horn conditional knowledge base KB and a literal-Horn default d such that $I \models \phi_r$ holds for every maximum weight world I of C iff KB z^*-entails d (see Appendix). Informally, the idea behind our construction is as follows. For each Horn clause ϕ_i with weight 2^{c_i}, we introduce a default $d_{c_i,i}$. By additional Horn clauses and literal-Horn defaults in KB, we then ensure that $z^*(d_{c_i,i}) = 2^{c_i}$ and that $d_{c_i,i}$ is falsified by a model I of some conjunction of atoms ϕ iff $I \not\models \phi_i$. Moreover, we ensure that all the other defaults in KB are falsified by every model of ϕ. This means that the minimal falsifying models of ϕ are exactly the maximum weight assignments for C. Hence, it finally just remains to choose an appropriate literal ψ, such that the default $d = \phi \rightarrow \psi$ is z^*-entailed by KB iff ϕ_r holds in all minimal falsifying models of ϕ.

$\mathrm{FP}^{\mathrm{NP}}$-hardness of computing the z^*-ranking for KB follows from an adaptation (see Appendix).

4.2 CONDITIONAL ENTAILMENT

As for the Π_2^{P} upper bound, recall from the definitions that KB does not conditionally entail d iff there exist a priority ordering $\prec$ on D admissible with KB and a $\prec$-preferred model I of $L \cup \{\phi\}$ such that $I \not\models \psi$. This is checked by the nondeterministic algorithm **not-cond-entailment** (see Fig. 2). The crucial point there is Step 3, which exploits the following lemma (see Appendix):

Lemma 4.1 *A priority ordering $\prec$ on D is admissible with a conditional knowledge base $KB = (L, D)$ iff each $d \in D$ is tolerated by $D_d = D - \{d' \in D \mid d' \prec d\}$ under L.*

Algorithm not-cond-entailment

Input: Conditional knowledge base (L, D) and a default $\phi \rightarrow \psi$.
Output: "Yes" iff (L, D) does not conditionally entail $\phi \rightarrow \psi$.

1. Guess a subset $\prec$ of $D \times D$;
2. **if** $\prec$ is not irreflexive **or** $\prec$ is not transitive **then halt**; /* $\prec$ is not a priority ordering */
3. **for each** $d \in D$ **do**
   if $\neg$ (d is tolerated by $D - \{d' \in D \mid d' \prec d\}$ under L)
   then halt; /* $\prec$ is not admissible */
4. Guess $I \in \mathcal{I}_{At}$;
5. **if** $I \not\models L \cup \{\phi\}$ **or** $I \models \psi$ **then halt**; /* wrong guess */
6. **if** some $J \in \mathcal{I}_{At}$ exists s.t. $J \prec I$ **and** $J \models L \cup \{\phi\}$ **then halt** /* I is not preferred */
   else return "Yes".

Figure 2: Algorithm **not-cond-entailment**

It is easy to see that testing the condition in the lemma for a given $d \in D$ is in NP. Thus, all queries in Step 3 and the query in Step 6 (see Fig. 2) can be solved using an NP-oracle. Modulo these queries, each step in **not-cond-entailment** is polynomial. Hence, deciding whether KB does not conditionally entail d is in $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2^{\mathrm{P}}$. Therefore, deciding conditional entailment is in $\Pi_2^{\mathrm{P}} = \text{co-}\Sigma_2^{\mathrm{P}}$.

To show that this upper bound is tight, we give a polynomial transformation from the following canonical Π_2^{P}-complete problem [32]. Given a collection of clauses $\alpha_1, \ldots, \alpha_l$ on the atoms $y_1, \ldots, y_m, x_1, \ldots, x_n$, where $m, n \geq 1$, decide whether the quantified Boolean formula (QBF) $\Phi = \forall y_1 \ldots \forall y_m \exists x_1 \ldots \exists x_n \, (\alpha_1 \wedge \cdots \wedge \alpha_l)$ evaluates to true.

We construct a literal-Horn $KB = (L, D)$ and a literal-Horn default $d = \phi \rightarrow \psi$ such that Φ evaluates to true iff KB conditionally entails d (see Appendix). Roughly speaking, it solves two major problems. Firstly, expressing the validity of a QBF. This is done exploiting the preference ordering on worlds. Secondly, transforming the clauses $\alpha_1, \ldots, \alpha_l$ into Horn clauses $\alpha_1^\star, \ldots, \alpha_l^\star$. For that, all positive literals y_i and x_j in $\alpha_1, \ldots, \alpha_l$ are replaced by new negative literals $\neg y_i'$ and $\neg x_j'$, respectively. We use L to express $y_i \wedge y_i' \Rightarrow \bot$ and $x_j \wedge x_j' \Rightarrow \bot$, and the preference ordering on worlds to express $\top \Rightarrow y_i \vee y_i'$ and $\top \Rightarrow x_j \vee x_j'$; thus, $y_i \equiv \neg y_i'$ and $x_j \equiv \neg x_j'$.

5 SUMMARY AND CONCLUSION

In this paper, we have established a sharp picture of the complexity of major approaches to default reasoning from conditional knowledge bases.

Several issues remains for further work. One issue is a fine-grained analysis of the effect of preprocessing and fixing parameters in the input. In particular, the amenability of the various semantics to compilability according to the frameworks proposed in [10, 23], and the effect of fixed parameters in the input [13, 30] seem to be worthwhile to study. Another issue is to identify tractable classes for the various approaches. Our results on literal-Horn knowledge bases suggest that for the "hard" semantics, severe further restrictions have to be imposed to gain tractability. Finally, other approaches to default reasoning (e.g., the recent one in [6]) may be analyzed from a complexity point of view.

APPENDIX: SELECTED PROOFS

Detailed proofs of all results are given in the full paper [17].

A MAXIMUM ENTROPY SEMANTICS

Lemma A.1 *Let $KB = (L, D)$ be ε-consistent with positive strength assignment σ. Let $s = \max(\{\sigma(d) \mid d \in D\})$. Let $z_s^\star(d_1), \ldots, z_s^\star(d_l)$, with $l \leq |D|$, be the sequence of default ranks computed in algorithm $z_s^\star$-ranking. Then $|z_s^\star(d_i)| \leq s \cdot 2^{i-1}$, for $1 \leq i \leq l$.*

Proof. The proof is by induction on $i = 1, \ldots, l$. Let $n = |D|$ be the cardinality of D.

Basis: For $i = 1$, we get $z_s^\star(d_1) = \sigma(d_1) \leq s$. *Induction:* Let $i > 1$. By the induction hypothesis, $|z_s^\star(d_j)| \leq s \cdot 2^{j-1}$ for all $j = 1, \ldots, i-1$. Hence, we get

$$|z_s^\star(d_i)| \leq \sigma(d_i) + |\mathrm{minv}(d_i) - \mathrm{minf}(d_i)| \leq$$
$$s + \textstyle\sum_{j=1}^{i-1} s \cdot 2^{j-1} = s + s \cdot (2^{i-1} - 1)) = s \cdot 2^{i-1}. \; \square$$

Theorem A.2 *Given a literal-Horn conditional KB that is ε-consistent and minimal-core, and a literal-Horn default d, deciding whether KB $z^\star$-entails d is P^{NP}-hard.*

Proof. We give a reduction from the P^{NP}-complete problem on weighted Horn clauses $C = \{\phi_i \mid 1 \leq i \leq m\}$ in Section 4.1. Suppose $\phi_i = \alpha_i \Rightarrow \beta_i$ and recall it has weight $w_i = 2^{c_i}$.

The set of atoms is $At = \{x_1, \ldots, x_n\} \cup A \cup B \cup T$, where $A = \{a_{i,j} \mid 1 \leq j \leq m, \, 0 \leq i \leq c_j\}$, $B = \{b_{i,j} \mid 1 \leq j \leq m, \, 0 \leq i \leq c_j\}$, and $T = \{t_{i,j} \mid 1 \leq j \leq m, \, 0 \leq i \leq c_j - 1\}$.

Then, $KB = (L, D)$ is defined as follows:

$$\begin{aligned} L &= \bigcup\{L_{i,j} \mid 1 \le j \le m, 0 \le i \le c_j\} \cup \\ &\quad \{\phi_{i,j} \mid 1 \le j \le m, 0 \le i \le c_j\} \\ D &= \{d_{i,j} \mid 1 \le j \le m, 0 \le i \le c_j\}, \end{aligned}$$

where $L_{i,j}$, $\phi_{i,j}$, and $d_{i,j}$ are defined as follows:

$$\begin{aligned} L_{i,j} &= \{b_{i,j} \Rightarrow a_{0,j}, b_{i,j} \Rightarrow t_{0,j}, \ldots, \\ &\quad\ b_{i,j} \Rightarrow a_{i-1,j}, b_{i,j} \Rightarrow t_{i-1,j}\} \\ \phi_{i,j} &= \begin{cases} b_{i,j} \wedge t_{i,j} \Rightarrow \bot & \text{if } i < c_j \\ b_{i,j} \wedge \alpha_i \Rightarrow \beta_i & \text{if } i = c_j \end{cases} \\ d_{i,j} &= a_{i,j} \rightarrow b_{i,j}\,. \end{aligned}$$

Now let the literal-Horn default $d = \phi \rightarrow \psi$ be defined by:

$$\phi = \bigwedge_{p \in A \cup T} p, \qquad \psi = b_{c_k,k}\,.$$

We prove the following:

- KB is ε-consistent. This is shown using the fact (essentially proved in [25, 28]) that any $KB = (L, D)$ is ε-consistent iff there exists an ordered partition $(D_0, \ldots, D_k)$ of D such that each default in D_i is tolerated under L by $\bigcup_{j=i}^{k} D_j$.

  Let $(D_0, \ldots, D_k)$ be defined by $k = \max(c_1, \ldots, c_m)$ and $D_i = \{d_{i,j} \mid d_{i,j} \in D\}$ for all $i = 1, \ldots, k$. Consider any $d_{i,j} \in D_i$. Let I be a world such that $I \models \alpha_j \Rightarrow \beta_j$, $I \models a_{l,j}$, for all $l \le i$, $I \models b_{i,j}$, $I \models t_{l,j}$, for all $l \le i - 1$, and I falsifies all remaining atoms $a_{i',j'}$, $b_{i',j'}$, and $t_{i',j'}$. Clearly such a world I exists. As easily seen, I verifies $d_{i,j}$ and $I \models L \cup \bigcup_{l=i}^{k} D_l$. Hence, $d_{i,j}$ is tolerated under L by $\bigcup_{l=i}^{k} D_l$. Thus, KB is ε-consistent.

- KB is minimal-core. Indeed, the world I such that $I \models a_{i,j}$ and $I \not\models p$ for any other atom p, falsifies the default $d_{i,j}$ while it satisfies $L \cup (D - \{d_{i,j}\})$.

- $z^*(d_{i,j}) = 2^i$, for all $d_{i,j} \in D$ (thus, $z^*(d_{c_j,j}) = 2^{c_j}$ for $j \le m$). Straightforward by induction.

- We finally show that $I \models \alpha_r \Rightarrow \beta_r$ holds in every I with maximum weight $\sum_{I \models \alpha_i \Rightarrow \beta_i} w_i$ iff KB z^*-entails d. We need some preparation as follows.

Let I be any world such that: (i) $I \models L \cup \{\phi\}$, (ii) I is a maximum weight world of C, and (iii) $I \models b_{c_j,j}$ iff $I \models \alpha_j \Rightarrow \beta_j$. Let I' be any world such that (i) holds but either (ii) or (iii) does not hold. We show that $\kappa^*(I) < \kappa^*(I')$. As easily seen, I and I' falsify all defaults $d_{i,j}$, for $j \le m$ and $i \le c_j - 1$. Let I'' result from I' by redefining $b_{c_j,j}$ to $I'' \models b_{c_j,j}$ iff $I' \models \alpha_j \Rightarrow \beta_j$, for $j \le n$. Then, $I'' \models L$, and no default $d_{i,j}$ satisfied by I'' is violated by $d_{i,j}$. Hence, it follows $\kappa^*(I'') \le \kappa^*(I')$. Furthermore, I and I'' satisfy $d_{c_j,j}$ iff they satisfy $\alpha_j \Rightarrow \beta_j$, for $j \le m$. Hence,

$$\begin{aligned} \kappa^*(I) &= \textstyle\sum_{d \in D} z^*(d) - \sum_{d \in D, I \models d} z^*(d) \\ &= \textstyle\sum_{d \in D} z^*(d) - \sum_{j \in \{1,\ldots,m\}, I \models \alpha_j \Rightarrow \beta_j} 2^{c_j} \\ &\le \textstyle\sum_{d \in D} z^*(d) - \sum_{j \in \{1,\ldots,m\}, I'' \models \alpha_j \Rightarrow \beta_j} 2^{c_j} \\ &= \textstyle\sum_{d \in D} z^*(d) - \sum_{d \in D, I'' \models d} z^*(d) \\ &= \kappa^*(I'') \end{aligned}$$

It follows $\kappa^*(I) \le \kappa^*(I'') \le \kappa^*(I')$. Moreover, since (ii) and (iii) hold for I, while either (ii) or (iii) does not hold for I', either $\kappa^*(I) < \kappa^*(I'')$ or $\kappa^*(I'') < \kappa^*(I')$, and thus $\kappa^*(I) < \kappa^*(I')$.

Assume first that $I \models \alpha_r \Rightarrow \beta_r$ holds for every maximum weight world I of C. Consider any such I. Hence, there exists a world I' such that $I' \models x_i$ iff $I \models x_i$, $I' \models b_{c_j,j}$ iff $I \models \alpha_j \Rightarrow \beta_j$, and $I' \models L \cup \{\phi \wedge \psi\}$. Moreover, there is no maximum weight world I'' of C such that $I'' \models L \cup \{\phi \wedge \neg\psi\}$ and $I'' \models b_{c_j,j}$ iff $I'' \models \alpha_j \Rightarrow \beta_j$. It follows $\kappa^*(\phi \wedge \psi) < \kappa^*(\phi \wedge \neg\psi)$, and thus KB z^*-entails d.

Assume next that $I \not\models \alpha_r \Rightarrow \beta_r$ for some maximum weight world of C. Hence, there exists a world I' such that $I' \models x_i$ iff $I \models x_i$, $I' \models b_{c_j,j}$ iff $I \models \alpha_j \Rightarrow \beta_j$, and $I' \models L \cup \{\phi \wedge \neg\psi\}$. It follows $\kappa^*(\phi \wedge \psi) \ge \kappa^*(\phi \wedge \neg\psi)$, and thus KB does not z^*-entail d. □

Theorem A.3 *The problem of computing the default ranking z^* for an ε-consistent minimal-core literal-Horn conditional knowledge base is* $\mathrm{FP}^{\mathrm{NP}}$*-hard.*

Proof. We give a polynomial transformation from a suitable variant of the problem used in the reduction in the proof of Theorem A.2, which is $\mathrm{FP}^{\mathrm{NP}}$-complete: Given C as there, compute the weight w of a maximum weight world I of C.

We slightly extend KB in the proof of Theorem A.2 as follows. We introduce new atoms a^* and b^*, and the following set of literal-Horn clauses L^* and literal-Horn default d^*:

$$\begin{aligned} L^* &= \{b^* \Rightarrow a_{i,j} \mid 1 \le j \le m, 0 \le i \le c_j\} \cup \\ &\quad \{b^* \Rightarrow t_{i,j} \mid 1 \le j \le m, 0 \le i < c_j\} \\ d^* &= a^* \rightarrow b^*\,. \end{aligned}$$

By similar arguments as in the proof of Theorem A.2, it is easy to see that $KB' = (L \cup L^*, D \cup \{d^*\})$ is ε-consistent and minimal-core. Moreover, its ranking z^* assigns all defaults $d_{i,j}$ the value 2^i and the default d^* the value $\sum_{j=1}^{m} \sum_{i=0}^{c_j} 2^i - w$. Hence, the weight w of a maximum weight world of C is given by $w = 2 \cdot \sum_{j=1}^{m} 2^{c_j} - m - z^*(d^*)$, which can be easily computed from z^*. □

B CONDITIONAL ENTAILMENT

Proof of Lemma 4.1. Assume that $\prec$ is admissible with KB, and let $d \in D$. Admissibility of $\prec$ implies that D_d is under L not in conflict with d (i.e., it tolerates d under L).

Conversely, assume every $d \in D$ is tolerated under L by D_d. Suppose that $\prec$ is not admissible with KB, i.e., some $D' \subseteq D$ is under L in conflict with some $d \in D$, and D' contains no default d' with $d' \prec d$. Hence, $D' \subseteq D_d$. Since D_d tolerates d under L, also D' tolerates d under L. But this contradicts that D' is under L in conflict with d. Hence, $\prec$ is admissible with KB. □

Theorem B.4 *Given a literal-Horn conditional knowledge base $KB = (L, D)$ and a literal-Horn default $d = \phi \rightarrow \psi$, deciding whether KB conditionally entails d is Π_2^{P}-hard.*

Proof. (Sketch) We build a literal-Horn $KB = (L, D)$ and a literal-Horn default $d = \phi \rightarrow \psi$ such that the QBF Φ from Section 4.2 evaluates to true iff KB conditionally entails d.

Let $At = A_y \cup A_x \cup \{a\}$, where $A_x = \{a_i, b_i, y_i, y_i' \mid 1 \leq i \leq m\}$, $A_x = \{c_j, d_j, e_j, f_j, x_j, x_j' \mid 1 \leq j \leq n\}$. We define $KB = (L, D)$ as follows:

$$L = L_1 \cup L_2 \cup L_3 \cup L_4, \quad D = \bigcup_{i=1}^{m} D_{1,i} \cup \bigcup_{j=1}^{n} D_{2,j},$$

where the sets of Horn clauses L_i, $i = 1, 2, 3, 4$ and the default sets D_i^1, D_j^2 are defined as follows:

$$\begin{aligned} L_1 &= \{\alpha_1^* \vee \neg a, \ldots, \alpha_l^* \vee \neg a\}, \\ L_2 &= \{\neg y_i \vee \neg y_i' \vee \neg a \mid i = 1, \ldots, m\}, \\ L_3 &= \{\neg x_j \vee \neg x_j' \vee \neg a \mid j = 1, \ldots, n\}, \\ L_4 &= \{x_j \Rightarrow \neg f_k,\ x_j' \Rightarrow \neg f_k \mid j, k = 1, \ldots, n\}, \end{aligned}$$

where $\alpha_1^*, \ldots, \alpha_l^*$ is obtained from $\alpha_1, \ldots, \alpha_l$ by replacing the positive literals y_i and x_j by the negative literals $\neg y_i'$ and $\neg x_j'$, respectively; and

$$\begin{aligned} D_{1,i} &= \{a_i \rightarrow y_i,\ b_i \rightarrow y_i'\}, \\ D_{2,j} &= \{c_j \wedge d_j \rightarrow x_j,\ c_j \wedge e_j \rightarrow x_j',\ c_j \rightarrow f_j\}. \end{aligned}$$

Finally, the default $d = \phi \rightarrow \psi$ is defined by

$$\begin{aligned} \phi &= a \wedge \Big(\bigwedge_{i=1}^{m} (a_i \wedge b_i)\Big) \wedge \Big(\bigwedge_{j=1}^{n} (c_j \wedge d_j \wedge e_j)\Big), \\ \psi &= \neg f_1. \end{aligned}$$

The set of all defaults that are conditionally entailed by KB is defined with respect to all priority orderings on D that are admissible with KB. By the following lemma, we can restrict our attention to all minimal priority orderings $\prec$ on D admissible with KB, where $\prec$ is smaller than $\prec'$ iff $\{(I, J) \mid I \prec J\}$ is a proper subset of $\{(I, J) \mid I \prec' J\}$.

Lemma (Geffner [20]). *A default $\phi \rightarrow \psi$ is conditionally entailed by a conditional knowledge base $KB = (L, D)$ iff ψ is satisfied in every $\prec$-preferred model of $L \cup \{\phi\}$ of every minimal priority ordering $\prec$ admissible with KB.* ◇

We note that every priority ordering $\prec$ on D admissible with KB contains the following pairs:

$$\begin{aligned} &c_j \rightarrow f_j \prec c_j \wedge d_j \rightarrow x_j \\ &\text{and } c_j \rightarrow f_j \prec c_j \wedge e_j \rightarrow x_j', \text{ for } j = 1, \ldots, n. \end{aligned} \tag{7}$$

Indeed, each set $\{c_j \rightarrow f_j\}$ tolerates under L neither $c_j \wedge d_j \rightarrow x_j$ nor $c_j \wedge e_j \rightarrow x_j'$.

Let $\prec^*$ contain exactly all pairs in (7). By Lemma 4.1, it follows that $\prec^*$ is admissible with KB. That is, $\prec^*$ is the least priority ordering on D admissible with KB. Thus, by Geffner's lemma, KB conditionally entails $d = \phi \rightarrow \psi$ iff $I \models \psi$ for every $\prec^*$-preferred model I of $L \cup \{\phi\}$.

We are now ready to show that Φ evaluates to true iff KB conditionally entails d.

($\Leftarrow$) Assume that Φ evaluates to false. Hence, a mapping $f : \{y_1, \ldots, y_m\} \rightarrow \{\bot, \top\}$ exists such that $\alpha = (\alpha_1 \wedge \cdots \wedge \alpha_l)[y_1/f(y_1), \ldots, y_m/f(y_m)]$ is unsatisfiable. Let I be such that (i) $I(y_i) = I(\neg y_i') =$ **true** iff $f(y_i) = \top$, for all $i \leq m$; (ii) $I(x_j) = I(x_j') =$ **false**, for all $j \leq n$; and (iii) $I(p) =$ **true** for any other atom p. Since each α_i^* contains at least one negative literal from $\{\neg x_j,\ \neg x_j' \mid 1 \leq j \leq n\}$, we have $I \models L_1$. Clearly, $I \models L_2 \cup L_3$, and also $I \models L_4$, $I \models \phi$, and $I \models \neg\psi$. Hence, $I \models L \cup \{\phi, \neg\psi\}$. It can be shown [17] that no world J exists such that $J \models L \cup \{\phi\}$ and $J \prec^* I$. Hence, I is a $\prec^*$-preferred model of $L \cup \{\phi\}$. Thus, KB does not conditionally entail d.

($\Rightarrow$) Assume that KB does not conditionally entail d. Thus, a $\prec^*$-preferred model I of $L \cup \{\phi\}$ exists such that $I \not\models \psi$, which means $I \models f_1$. The clauses in L_4 imply that $I(x_j) = I(x_j') =$ **false**, for all $j \leq n$; preferredness of I implies that $I(f_j) =$ **true** (as $c_j \rightarrow f_j$ will be satisfied), for all $j \leq n$. Let f: $\{y_1, \ldots, y_m\} \rightarrow \{\bot, \top\}$ be defined by $f(y_i) = \top$ iff $I \models a_i \rightarrow y_i$. We show that $\alpha = (\alpha_1 \wedge \cdots \wedge \alpha_l)[y_1/f(y_1), \ldots, y_m/f(y_m)]$ is unsatisfiable. Towards a contradiction, suppose some world I' on $x_1, \ldots, x_n$ exists such that $I' \models \alpha$. Let I'' coincide on all y_i, y_i' with I and such that $I''(x_j) = I''(\neg x_j') = I'(x_j)$; $I''(f_j) =$ **false**, for all $j \leq n$; and $I''(p) =$ **true** for every other atom p. Then $I'' \models L \cup \{\phi\}$ and $I'' \prec^* I$. Hence, I is not a $\prec^*$-preferred model of $L \cup \{\phi\}$, which is a contradiction. Consequently, α is unsatisfiable, and thus Φ evaluates to false. □

Acknowledgements

This work has been partially supported by the Austrian Science Fund Project N Z29-INF and a DFG grant.

References

[1] E. W. Adams. *The Logic of Conditionals*, volume 86 of *Synthese Library*. D. Reidel, Dordrecht, Netherlands, 1975.

[2] R. Ben-Eliyahu. NP-complete problems in optimal horn clauses satisfiability. Technical Report R-158, UCLA, Computer Science Dept., 1991.

[3] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, and H. Prade. Inconsistency management and prioritized syntax-based entailment. In *Proceedings IJCAI-93*, pages 640–645. 1993.

[4] S. Benferhat, D. Dubois, and H. Prade. Representing default rules in possibilistic logic. In *Proceedings KR'92*, pages 673–684. 1992.

[5] S. Benferhat, D. Dubois, and H. Prade. Nonmonotonic reasoning, conditional objects and possibility theory. *Artif. Intell.*, 92(1–2):259–276, 1997.

[6] S. Benferhat, A. Saffiotti, and P. Smets. Belief functions and default reasoning. In *Proceedings UAI'95*, pages 19–26. 1995.

[7] R. A. Bourne and S. Parsons. Maximum entropy and variable strength defaults. In *Proceedings IJCAI-99*, pages 50–55. 1999.

[8] C. Boutilier. Conditional logics of normality as modal systems. In *Proceedings AAAI-90*, pages 594–599, 1990.

[9] M. Cadoli. The complexity of model checking for circumscriptive formulae. *Inf. Process. Lett.*, 44:113–118, 1992.

[10] M. Cadoli, F. M. Donini, P. Liberatore, and M. Schaerf. Feasibility and unfeasibility of off-line processing. In *Proceedings ISTCS'96*, pages 100–109. 1996.

[11] M. Cadoli and M. Lenzerini. The complexity of propositional closed world reasoning and circumscription. *J. Comput. Syst. Sci.*, 43:165–211, 1994.

[12] C. Cayrol, M.-C. Lagasquie-Schiex, and T. Schiex. Nonmonotonic reasoning: From complexity to algorithms. *Ann. Math. Artif. Intell.*, 22(3–4):207–236, 1998.

[13] R. G. Downey, M. F. Fellows, and M. R. Fellows. *Parameterized Complexity*. Springer, 1998.

[14] D. Dubois and H. Prade. Possibilistic logic, preferential models, non-monotonicity and related issues. In *Proceedings IJCAI-91*, pages 419–424. 1991.

[15] D. Dubois and H. Prade. Conditional objects as nonmonotonic consequence relationships. *IEEE Trans. Syst. Man Cybern.*, 24:1724–1740, 1994.

[16] T. Eiter and G. Gottlob. Propositional circumscription and extended closed world reasoning are Π_2^P-complete. *Theor. Comput. Sci.*, 114(2):231–245, 1993. Addendum 118:315.

[17] T. Eiter and T. Lukasiewicz. Complexity results for default reasoning from conditional knowledge bases. Technical Report INFSYS RR-1843-99-10, Institut für Informationssysteme, TU Wien, 1999.

[18] N. Friedman and J. Y. Halpern. Plausibility measures and default reasoning. *J. ACM*. To appear.

[19] N. Friedman and J. Y. Halpern. On the complexity of conditional logics. In *Proceedings KR'94*, pages 202–213. 1994.

[20] H. Geffner. *Default Reasoning: Causal and Conditional Theories*. MIT Press, Cambridge, MA, 1992.

[21] H. Geffner. High probabilities, model preference and default arguments. *Mind and Machines*, 2:51–70, 1992.

[22] H. Geffner and J. Pearl. Conditional entailment: Bridging two approaches to default reasoning. *Artif. Intell.*, 53(2–3): 209–244, 1992.

[23] G. Gogic, H. Kautz, C. H. Papadimitriou, and B. Selman. The comparative linguistics of knowledge representation. In *Proceedings IJCAI-95*, pages 862–869. 1995.

[24] M. Goldszmidt, P. Morris, and J. Pearl. A maximum entropy approach to nonmonotonic reasoning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(3):220–232, 1993.

[25] M. Goldszmidt and J. Pearl. On the consistency of defeasible databases. *Artif. Intell.*, 52(2):121–149, 1991.

[26] M. Goldszmidt and J. Pearl. System Z^+: A formalism for reasoning with variable strength defaults. In *Proceedings AAAI-91*, pages 399–404. 1991.

[27] M. Goldszmidt and J. Pearl. Rank-based systems: A simple approach to belief revision, belief update and reasoning about evidence and actions. In *Proceedings KR'92*, pages 661–672. 1992.

[28] M. Goldszmidt and J. Pearl. Qualitative probabilities for default reasoning, belief revision, and causal modeling. *Artif. Intell.*, 84(1–2):57–112, 1996.

[29] G. Gottlob. Complexity results for nonmonotonic logics. *J. Log. Comput.*, 2(3):397–425, June 1992.

[30] G. Gottlob, F. Scarcello, and M. Sideri. Fixed parameter complexity in AI and nonmonotonic reasoning. In *Proceedings LPNMR'99*, pages 1–18, 1999.

[31] B. Jenner and J. Toran. Computing functions with parallel queries to NP. *Theor. Comput. Sci.*, 141:175–193, 1995.

[32] D. S. Johnson. A catalog of complexity classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 2, pages 67–161. MIT Press, Cambridge, MA, 1990.

[33] H. Kautz and B. Selman. Hard problems for simple default logics. *Artif. Intell.*, 49:243–279, 1991.

[34] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artif. Intell.*, 14(1):167–207, 1990.

[35] M. W. Krentel. The complexity of optimization problems. *J. Comput. Syst. Sci.*, 36(3):490–509, 1988.

[36] P. Lamarre. A promenade from monotonicity to nonmonotonicity following a theorem prover. In *Proceedings KR'92*, pages 572–580. 1992.

[37] D. Lehmann. What does a conditional knowledge base entail? In *Proceedings KR'89*, pages 212–222. 1989.

[38] D. Lehmann. Another perspective on default reasoning. *Ann. Math. Artif. Intell.*, 15(1):61–82, 1995.

[39] D. Lehmann and M. Magidor. What does a conditional knowledge base entail? *Artif. Intell.*, 55(1):1–60, 1992.

[40] B. Nebel. How hard is it to revise a belief base? In D. M. Gabbay and P. Smets, editors, *Handbook on Defeasible Reasoning and Uncertainty Management Systems*, volume 3: Belief Change, pages 77–145. Kluwer Academic, Dordrecht, Netherlands, 1998.

[41] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.

[42] J. Pearl. Probabilistic semantics for nonmontonic reasoning: A survey. In *Proceedings KR'89*, pages 505–516. 1989.

[43] J. Pearl. System Z: A natural ordering of defaults with tractable applications to default reasoning. In *Proceedings TARK'90*, pages 121–135. 1990.

[44] A. Selman. A taxonomy of complexity classes of functions. *J. Comput. Syst. Sci.*, 48:357–381, 1994.

[45] Y. Shoham. A semantical approach to nonmonotonic logics. In *Proceedings of the 2nd IEEE Symposium on Logic in Computer Science*, pages 275–279, 1987.

[46] W. Spohn. Ordinal conditional functions: A dynamic theory of epistemic states. In W. Harper and B. Skyrms, editors, *Causation in Decision, Belief Change, and Statistics*, volume 2, pages 105–134. D. Reidel, Dordrecht, 1988.

[47] J. Stillman. It's not my default: The complexity of membership problems in restricted propositional default logic. In *Proceedings AAAI-90*, pages 571–579, 1990.

[48] J. Stillman. The complexity of propositional default logic. In *Proceedings AAAI-92*, pages 794–799, 1992.

Uniform semantic treatment of default and autoepistemic logics

Marc Denecker
Department of Computer Science
K.U.Leeuven
Celestijnenlaan 200A, B-3001 Heverlee
Belgium

Victor W. Marek
Computer Science Department
University of Kentucky
Lexington, KY 40506-0046
USA

Mirosław Truszczyński
Computer Science Department
University of Kentucky
Lexington, KY 40506-0046
USA

Abstract

We revisit the issue of connections between two leading formalisms in nonmonotonic reasoning: autoepistemic logic and default logic. For each logic we develop a comprehensive semantic framework based on the notion of a *belief pair*. The set of all belief pairs together with the so called *knowledge ordering* forms a complete lattice. For each logic, we introduce several semantics by means of fixpoints of operators on the lattice of belief pairs. Our results elucidate an underlying isomorphism of the respective semantic constructions. In particular, we show that the interpretation of defaults as modal formulas proposed by Konolige allows us to represent all semantics for default logic in terms of the corresponding semantics for autoepistemic logic. Thus, our results conclusively establish that default logic can indeed be viewed as a fragment of autoepistemic logic. However, as we also demonstrate, the semantics of Moore and Reiter are given by *different* operators and occupy *different* locations in their corresponding families of semantics! This result explains the source of the longstanding difficulty to formally relate these two semantics. In the paper, we also discuss approximating skeptical reasoning with autoepistemic and default logics and establish constructive principles behind such approximations.

1 INTRODUCTION

Due to their applications in knowledge representation and, more specifically, in commonsense reasoning, abduction, diagnosis, belief revision, planning and reasoning about action, default and autoepistemic logics are among the most extensively studied nonmonotonic formalisms[1]. Still, after almost two decades of research in the area several key questions remain open.

The first of them is the question of the relationship between default and autoepistemic logics. Default logic was introduced by Reiter [Rei80] to formalize reasoning about defaults, that is, statements that describe what *normally* is the case, in the absence of contradicting information. Autoepistemic logic was proposed by Moore [Moo84] to describe the belief states of rational agents reflecting upon their own beliefs and disbeliefs. Although the motivation and syntax of both logics are different, it has been clear for a long time that they are closely related. However, despite much work [Kon88, MT89a, Tru91, MT93, Got95] no truly satisfactory account of the relationship was found. Konolige [Kon88] related default logic to a version of autoepistemic logic based on the notion of a *strongly grounded expansion* — a concept that depends on a syntactic representation of a theory. Marek and Truszczynski related default logic to two modal nonmonotonic logics related but different from autoepistemic logic: the nonmonotonic modal logic **N** [MT89a] and nonmonotonic modal logic **S4F** [Tru91]. Finally, Gottlob [Got95] found a relationship between default and autoepistemic logics but the translation he used was not modular. In fact, he proved that a modular translation of default logic into autoepistemic logic does not exist. These results seem to point to some misalignment between extensions of default theories and expansions of modal theories. Our results in this paper finally clarify the picture.

Another problem is related to the fixpoint definitions of extensions (in the case of default logic) and expansions (in the case of autoepistemic logic). They

[1] For a detailed discussion of these two formalisms and of their applications, and for additional references, the reader is referred to [MT93].

provide no insights into constructive processes agents might use to build their belief sets based on default or modal theories describing base facts. Finally, there is a problem of high computational complexity of reasoning with extensions and expansions. The problems to decide the existence of an extension (expansion) is Σ_P^2-complete, the problem to compute the intersection of extensions (expansions) — it is needed for skeptical reasoning — is Π_P^2-hard.

In this paper we develop a unifying semantic treatment of default and autoepistemic logics and use it to address the three issues discussed above. For each logic we define a family of 2-, 3- and 4-valued semantics and show that they include all major semantics for default and autoepistemic logics. Within our framework we define semantics that generalize Kripke-Kleene and well-founded semantics for logic programs. These semantics allow us to approximate skeptical default and autoepistemic reasoning and they can be computed faster than extensions and expansions (assuming that the polynomial hierarchy does not collapse). Most importantly, we show that the unified semantic picture of default and autoepistemic logics described here allows us to pinpoint the exact nature of how they are related.

Our approach is motivated by the algebraic approach proposed by Fitting in his analysis of semantics for logic programs with negation [Fit99], and extends our earlier work on 3-valued semantics for autoepistemic logics [DMT98]. It relies on the concept of a *belief pair*, a pair (P, S), where P and S are sets of 2-valued interpretations. The concept of a belief pair generalizes the notion of a *possible-world structure*, that is, a set of 2-valued interpretations, often used to define the semantics of the modal logic S5 and also used by Moore and Levesque [Moo84, Lev90] in their work on autoepistemic logic. Specifically, each possible-world structure Q can be identified with a belief pair (Q, Q). Belief pairs allow us to approximate the state of beliefs of an agent whose beliefs are represented by a possible-world structure.

Given a possible-world structure Q or a belief pair B, it is often possible to describe a way in which Q (or B) could be revised to more accurately reflect the agent's beliefs. Such a revision procedure can be formally described by an operator. Fixpoints of such operators are often used to specify semantics of nonmonotonic formalisms as they represent those belief states of the agent that cannot be revised away. Sometimes fixpoints that satisfy some minimality conditions are additionally distinguished.

These intuitions underlie the original definition of an *expansion* of a modal theory T as a fixpoint of a certain operator D_T on the set of all possible-world structures [Moo84]. In [DMT98], it is shown that the operator D_T can be extended to the set of belief pairs. The resulting operator, $\mathcal{D}_T$, yields a multi-valued generalization of expansions and a semantics that approximates skeptical autoepistemic reasoning. We refer to it as the Kripke-Kleene semantics as it generalizes Kripke-Kleene semantics for logic programs. In this paper, we derive from the operator $\mathcal{D}_T$ two new operators and show that their fixpoints give rise to semantics for autoepistemic theories under which circular dependence of beliefs upon themselves, present in autoepistemic logic of Moore, is eliminated. One of these semantics is shown to correspond to Reiter's semantics of default logic. The other one can be viewed as a generalization of the well-founded semantics.

While possible-world semantics were used in the study of autoepistemic logic, they had only a marginal effect on the development of default logic. In this paper, we show that possible-world semantics approach can be extended to the case of default logic. Namely, in a close analogy with autoepistemic logic, for every default theory Δ, we introduce an operator $\mathcal{E}_\Delta$ defined on the set of belief pairs. We show that the operator $\mathcal{E}_\Delta$ gives rise to three other operators and that the fixpoints of these operators yield several semantics for default theories. Among these semantics are the semantics of extensions by Reiter, the stationary semantics [PP94], the well-founded semantics for default logic [BS91] (it approximates skeptical reasoning under extensions), the semantics of weak extensions [MT89a] and the Kripke-Kleene semantics (it approximates skeptical reasoning under weak extensions).

Our results settle the issue of the relationship between autoepistemic and default logics. We show that the operators $\mathcal{E}_\Delta$ and $\mathcal{D}_T$ are closely related if a default theory Δ is interpreted as a modal theory T given by the translation proposed by Konolige [Kon88]. Our results show that under the Konolige's translation, the *families* of semantics for default and autoepistemic logics are isomorphic and default logic can be viewed, as has long been expected, as a fragment of autoepistemic logic. But this correspondence does not relate extensions and expansions! The semantics corresponding to these concepts occupy different locations in their respective families of semantics and have different properties.

We also point out that the Kripke-Kleene and well-founded semantics studied in the paper have better computational properties than the semantics of expansions and extensions. We conclude with comments

about further generalizations and open problems.

2 PRELIMINARIES

The formal language for the semantic study developed in this paper is that of lattices, operators and fixpoints. The key result is that by Tarski and Knaster [Tar55] stating that a monotone operator on a complete lattice has a least fixpoint.

By At we denote the set of atoms of the propositional language under consideration. The set of all 2-valued interpretations of At will be denoted by $\mathcal{A}$. Any set $Q \subseteq \mathcal{A}$ is called a *possible-world* structure and can be viewed as a universal Kripke model [Che80]. Possible-world structures are a basic tool in semantic studies of modal logics. They were also used in the context of autoepistemic logic [Moo84, Lev90, MT93].

A collection of all possible-world structures will be denoted by $\mathcal{W}$. This set can be ordered by the reverse set inclusion: for $Q_1, Q_2 \in \mathcal{W}$, $Q_1 \sqsubseteq Q_2$ if $Q_2 \subseteq Q_1$ (the smaller the set of possible worlds, the bigger the theory it determines). Clearly, $\langle \mathcal{W}, \sqsubseteq \rangle$ is a complete lattice. To study formalisms based on the modal language, we define the *truth function* $\mathcal{H}_{Q,I}$ inductively as follows (Q is a possible-world structure, $I \in \mathcal{A}$ is an interpretation):

1. $\mathcal{H}_{Q,I}(p) = I(p)$, if p is an atom
2. Boolean connectives are handled in the standard Tarskian way
3. $\mathcal{H}_{Q,I}(K\varphi) = \mathbf{t}$, if for every interpretation $J \in Q$, $\mathcal{H}_{Q,J}(\varphi) = \mathbf{t}$, and $\mathcal{H}_{Q,I}(K\varphi) = \mathbf{f}$, otherwise.

The value of a modal atom $K\varphi$ given by $\mathcal{H}_{Q,I}$ does not depend on I. Thus, it is determined only by the possible-world structure in question. It is clear that the *meta-knowledge* specified by a possible-world structure Q is complete: all modal atoms $K\varphi$ are either true or false with respect to Q.

For every modal theory T, Moore [Moo84] defined an operator D_T on $\mathcal{W}$ by:

$$D_T(Q) = \{I : \mathcal{H}_{Q,I}(\varphi) = \mathbf{t}, \text{ for every } \varphi \in T\}.$$

Moore called the theory of a fixpoint of D_T an *expansion*[2].

In [DMT98], Moore's approach was extended to the 3-valued case, in which a possibility of incomplete meta-knowledge is admitted. A key concept is that of a *belief pair*, that is, a pair (P, S) of possible-world structures. In a belief pair (P, S), P can be viewed as a representation of a conservative (pessimistic) view on what is believed while S can be regarded as a representation of a liberal (gullible) view. If $S \subseteq P$, formulas believed in according to the conservative view captured by P, are believed in according to the liberal view represented by S. Consequently, a belief pair (P, S) such that $S \subseteq P$ is called *consistent*. However, the ways in which the agent establishes the estimates P and S may be independent of each other and, thus, we allow belief pairs (P, S) such that S is not a subset of P. We refer to them as *inconsistent*[3]. In applications we are mostly interested in consistent belief pairs. Constructive techniques for building belief pairs given a base theory, which are described in the paper, result in consistent belief pairs. However, admitting inconsistent belief pairs completes the picture, leads to simple intuitions behind mathematical arguments and results in more elegant algebraic structures.

With a belief pair (P, S) and a 2-valued interpretation I we associate a 2-valued truth function $\mathcal{H}^2_{(P,S),I}$ defined on a modal language. The idea is to define $\mathcal{H}^2_{(P,S),I}(\varphi)$ so that it provides a conservative estimate to the truth value of φ with respect to a belief pair (P, S). Since P represents a conservative point of view and S a liberal one, to get a conservative estimate we use P to evaluate positive occurrences of modal atoms and S to evaluate negative ones. The definition is inductive:

1. $\mathcal{H}^2_{(P,S),I}(p) = I(p)$, for every atom p
2. Conjunction and disjunction are treated in the standard Tarskian way
3. $\mathcal{H}^2_{(P,S),I}(\neg\varphi) = \neg\mathcal{H}^2_{(S,P),I}(\varphi)$
4. $\mathcal{H}^2_{(P,S),I}(K\varphi) = \mathbf{t}$ if $\mathcal{H}^2_{(P,S),J}(\varphi) = \mathbf{t}$ for all $J \in P$. $\mathcal{H}^2_{(P,S),I}(K\varphi) = \mathbf{f}$, otherwise.

We stress that when evaluating the negation of a formula the roles of P and S are switched, which ensures that modal literals appearing positively in a formula are evaluated with respect to P while those appearing negatively are evaluated with respect to S.

Clearly, to construct a liberal estimate for the truth value of φ with respect to a belief pair (P, S) we can proceed similarly and use S (P) to evaluate modal literals appearing positively (negatively) in φ. It is easy to see, however, that the resulting truth function

[2] In the paper, we will frequently use the same term to denote the fixpoint of an operator and its theory. So, fixpoints of D_T will be referred to as expansions, as well.

[3] In [DMT98] only consistent belief pairs were considered.

can be expressed as $\mathcal{H}^2_{(S,P),I}$ (we reverse the roles of P and S).

Conservative and liberal estimates of truth values of formulas can be combined into a single *4-valued* estimate. We say that the logical value of a formula φ is true, $\mathbf{t}_4$ (false, $\mathbf{f}_4$), if both conservative and liberal estimates of its truth value are equal to $\mathbf{t}$ ($\mathbf{f}$). We say that the logical value φ is *unknown*, $\mathbf{u}$, if the conservative estimate is $\mathbf{f}$ and the liberal estimate is $\mathbf{t}$. Finally, the logical value of φ is *inconsistent*, $\mathbf{i}$, if the conservative estimate is $\mathbf{t}$ and the liberal estimate is $\mathbf{f}$. Thus, the estimates given by $\mathcal{H}^2_{(P,S),I}(\varphi)$ and $\mathcal{H}^2_{(S,P),I}(\varphi)$ yield a 4-valued truth function

$$\mathcal{H}^4_{(P,S),I}(\varphi) = (\mathcal{H}^2_{(P,S),I}(\varphi), \mathcal{H}^2_{(S,P),I}(\varphi)),$$

where $\mathbf{t}_4 = (\mathbf{t},\mathbf{t})$, $\mathbf{f}_4 = (\mathbf{f},\mathbf{f})$, $\mathbf{u} = (\mathbf{f},\mathbf{t})$ and $\mathbf{i} = (\mathbf{t},\mathbf{f})$.

We define the *meta-knowledge* of a belief pair (P,S) as the set of all formulas φ such that both conservative and liberal estimates of the truth value of the epistemic atom $K\varphi$ coincide (are both true or are both false). Clearly, the meta-knowledge of a belief pair (P,S) need not be complete. For some modal atoms $K\varphi$ the two estimates may disagree ($K\varphi$ may be assigned value $\mathbf{u}$ or $\mathbf{i}$). However, it is easy to see, that for a complete belief pair (P,P), $\mathcal{H}^4_{(P,P),I} = \mathcal{H}_{P,I}$, for every interpretation $I \in \mathcal{A}$. In other words, belief pairs and the truth function $\mathcal{H}^4_{(P,S),I}$ generalize possible-world structures and the truth function $\mathcal{H}_{P,I}$. In addition, for a consistent belief pair (P,S), $\mathcal{H}^4_{(P,S),I}$ never assigns value $\mathbf{i}$ and coincides with the 3-valued truth function defined in [DMT98].

The set of all belief pairs is denoted by $\mathcal{B}$. It can be ordered by the *knowledge ordering* $\preceq_{kn}$: $(P_1,S_1) \preceq_{kn} (P_2,S_2)$ if $S_1 \subseteq S_2$, and $P_2 \subseteq P_1$. The set $\mathcal{B}$ with the ordering $\preceq_{kn}$ forms a complete lattice and, consequently, a $\preceq_{kn}$-monotone operator on $\mathcal{B}$ is guaranteed to have a least fixpoint by the result of Tarski and Knaster [Tar55]. The ordering $\preceq_{kn}$ coincides with the ordering of increasing meta-knowledge (decreasing meta-ignorance): $(P_1,S_1) \preceq_{kn} (P_2,S_2)$ if and only if the set of modal atoms with the same conservative and liberal estimates with respect to (P_1,S_1) is contained in the set of modal atoms for which conservative and liberal estimates with respect to (P_2,S_2) are the same.

Let (P,S) be a belief pair. Following [DMT98], we set

$$\mathcal{D}_T(P,S) = (\mathcal{D}^l_T(P,S), \mathcal{D}^u_T(P,S)),$$

where

$$\mathcal{D}^l_T(P,S) = \{I : \mathcal{H}^2_{(S,P),I}(T) = \mathbf{t}\}$$

and

$$\mathcal{D}^u_T(P,S) = \{I : \mathcal{H}^2_{(P,S),I}(T) = \mathbf{t}\}.$$

Fixpoints of the operator $\mathcal{D}_T$ will be called *partial expansions*.

Speaking intuitively, the operator $\mathcal{D}_T$ describes how an agent might revise a belief pair (P,S). The possible-world structure P is replaced by the structure P' consisting of those interpretations I for which all formulas from T are true according to the liberal estimates of truth values (given (P,S)). A liberal criterion for selecting possible worlds (interpretations) to P' results in a possible-world structure capturing a conservative point of view. By duality between conservative and liberal approaches, S can be replaced by S' consisting of all interpretations I such that all formulas from T are true according to the conservative estimates of truth values (given (P,S)).

Let us recall that D_T stands for the operator introduced by Moore. The operator $\mathcal{D}_T$ allows us to reconstruct the semantic approach to autoepistemic logic proposed by Moore.

Theorem 2.1 *Let T be a modal theory. Then, for every possible-world structure P, $\mathcal{D}_T(P,P) = (D_T(P), D_T(P))$. Consequently, a belief pair (P,P) is a fixpoint of $\mathcal{D}_T$ if and only if P is a fixpoint of D_T.*

Theorem 2.1 implies that there is a natural one-to-one correspondence between the complete partial expansions of T and the expansions of T. Thus, partial expansions (consistent partial expansions) can be viewed as 4-valued (3-valued) generalizations of Moore's expansions.

The key property of the operator $\mathcal{D}_T$ is its $\preceq_{kn}$-monotonicity. It follows that $\mathcal{D}_T$ has a unique $\preceq_{kn}$-least fixpoint. We denote it by $KK(T)$ and refer to it as the *Kripke-Kleene fixpoint* (or semantics) for T[4]. Kripke-Kleene fixpoint has a clear constructive flavor (it can be obtained by iterating the operator $\mathcal{D}_T$, starting at the least informative belief pair, $(\mathcal{A},\emptyset)$). It approximates all partial expansions and, in particular, approximates the skeptical reasoning with expansions.

Theorem 2.2 *Let T be a modal theory.*

1. *The fixpoint $KK(T)$ is consistent.*
2. *For every partial expansion B of T, $KK(T) \preceq_{kn} B$.*
3. *If $K\varphi$ is true with respect to $KK(T)$ then φ belongs to every expansion of T. If $K\varphi$ is false with*

[4]There is a close analogy between the least fixpoint of the operator $\mathcal{D}_T$ and the Kripke-Kleene semantics for logic programs.

respect to $KK(T)$ then φ belongs to no expansion of T.

Deciding the truth value of a modal atom $K\varphi$ with respect to the Kripke-Kleene fixpoint is in the class Δ_P^2 [DMT98]. Thus, unless the polynomial hierarchy collapses, it is a simpler problem than the problem of computing expansions or their intersection.

We can also use the Kripke-Kleene semantics as a test for the uniqueness of an expansion. Namely, we can first compute $KK(T)$ and check if $KK(T)$ is complete. If it is, T has a unique expansion. This method is computationally better (again, if the polynomial hierarchy does not collapse) than the straightforward one which computes all expansions of T. However, it is incomplete — there are theories T with a unique expansion and such that $KK(T)$ is not complete.

3 AUTOEPISTEMIC LOGIC

The operator $\mathcal{D}_T$ allows us to define two additional operators: the operator D_T^{st} defined on the lattice $\mathcal{W}$, and the operator $\mathcal{D}_T^{st}$ defined on the lattice $\mathcal{B}$. They give rise to new semantics for autoepistemic logic that are closely related to the semantics of extensions for default logic. One of them is a perfect match to Reiter's semantics of extensions for default logic, an object long sought after in the autoepistemic logic.

Let us recall that the operator $\mathcal{D}_T$ associates with each belief pair (P,S) its revised variant $(P',S') = \mathcal{D}_T(P,S)$. The way in which P' is obtained is described by the operator $\mathcal{D}_T^l$. Namely, $P' = \mathcal{D}_T^l(P,S)$. If we fix S, this operator becomes a monotone operator on $\mathcal{W}$ (it follows from the fact that $\mathcal{D}_T$ is $\preceq_{kn}$-monotone). Hence, its least fixpoint can be viewed as the preferred revision of P, given a fixed S. Let us define, then,

$$D_T^{st}(S) = lfp(\mathcal{D}_T^l(\cdot, S)).$$

Similarly, we can argue that $lfp(\mathcal{D}_T^u(P,\cdot))$ can be regarded as a preferred revision of S, given P. It turns out that $lfp(\mathcal{D}_T^u(P,\cdot)) = D_T^{st}(P)$. Thus, we define an operator $\mathcal{D}_T^{st}$ on belief pairs as follows:

$$\mathcal{D}_T^{st}(P,S) = (D_T^{st}(S), D_T^{st}(P)).$$

Clearly, D_T^{st} is an operator on $\mathcal{W}$. The fixpoints of the operator D_T^{st} (and also their theories) will be referred to as *extensions*. The choice of the term is not arbitrary. We show in Section 5 that extensions of modal theories can be regarded as generalizations of extensions of default theories. We have the following property relating fixpoints of the operators D_T^{st} and $\mathcal{D}_T^{st}$.

Theorem 3.1 *For every modal theory T, a possible-world structure P is a fixpoint of D_T^{st} if and only if a belief pair (P,P) is a fixpoint of $\mathcal{D}_T^{st}$.*

It follows that the semantics of fixpoints of $\mathcal{D}_T^{st}$ can be viewed as a 4-valued version of the semantics of extensions. Similarly, consistent fixpoints can be thought of as a 3-valued generalizations of extensions. Consequently, we refer to the fixpoints of the operator $\mathcal{D}_T^{st}$ as *partial extensions*.

The circular dependence allowing the agent to accept p to the belief set just on the basis of this agent believing in p, allowed under the semantics of expansions, is eliminated in the case of extensions.. For instance, the theory $\{Kp \Rightarrow p\}$ has two expansions. One of them is determined by the possible-world structure consisting of all interpretations, the other one — by the possible-world structure consisting of all interpretations in which p is true. It is this second expansion that suffers from circular-argument problem: the belief in p is the only justification for having p in this expansion. In the same time, the theory $\{Kp \Rightarrow p\}$ has exactly *one* extension, the one given by the possible-world structure consisting of all interpretations. The atom p is not true in it and, hence, circular arguments are not used in the construction of this expansion.

The operator D_T^{st} is antimonotone. It follows that the operator $\mathcal{D}_T^{st}(P,S)$ is $\preceq_{kn}$-monotone. Thus, by the result of Tarski and Knaster, it has the least fixpoint. We will denote this fixpoint by $WF(T)$ and refer to it as the *well-founded fixpoint* (or semantics) of T. Our choice of the term is again not accidental. This semantics is closely related to the well-founded semantics of default logic [BS91] and logic programming [VRS91]. We have the following result indicating that well-founded semantics can be used to approximate all partial extensions and, in particular, all extensions of a modal theory. It also shows that the well-founded semantics provides a sufficient condition for the uniqueness of an extension.

Theorem 3.2 *Let T be a modal theory and let $WF(T) = (P,S)$. Then:*

1. *The fixpoint $WF(T)$ is consistent.*
2. *For every partial extension B, $WF(T) \preceq_{kn} B$.*
3. *If $\mathcal{H}^4_{WF(T),I}(K\varphi) = \mathbf{t}$, then φ belongs to every extension of T. Similarly, if $\mathcal{H}^4_{WF(T),I}(K\varphi) = \mathbf{f}$, then φ does not belong to any extension.*
4. *If $WF(T)$ is complete (that is, $P = S$) then T has a unique extension corresponding to the possible-world structure P.*

Based on the approach developed in [DMT98] for computing the Kripke-Kleene semantics, we can establish computational properties of the well-founded semantics. We have the following result.

Theorem 3.3 *The problem of computing the well-founded semantics is the class* Δ_P^2.

Thus, assuming that the polynomial hierarchy does not collapse, computing the well-founded semantics of a theory T is easier than computing the intersection of extensions of T (that is, the set of skeptical consequences of T). Since the well-founded semantics of a modal theory T approximates all extensions, it can be used to speed up the computation of their intersection.

The next result connects expansions and the Kripke-Kleene semantics with extensions and the well-founded semantics. It shows that the well-founded semantics is stronger than the Kripke-Kleene semantics and that (partial) extensions of T are (partial) expansions of T satisfying some minimality condition.

Theorem 3.4 *Let T be a modal theory. Then:*

1. $KK(T) \preceq_{kn} WF(T)$.
2. *Every extension of T is a $\sqsubseteq$-minimal expansion of T.*
3. *Every partial extension (P, S) of T is a minimal partial expansion of T in the following sense: for every partial expansion (P', S'), if $P' \sqsubseteq P$ and $S' \sqsubseteq S$, then $P = P'$ and $S = S'$.*

We conclude this section with a schematic illustration of the panorama of semantics for autoepistemic logic. The central position is occupied by the operator $\mathcal{D}_T$. Its fixpoints yield the semantics of partial expansions and its least fixpoint yields the Kripke-Kleene semantics. Restriction of the operator $\mathcal{D}_T$ to complete belief pairs leads to the operator D_T, originally introduced by Moore, and results in the semantics of expansions. The operator $\mathcal{D}_T$ also gives rise to the operators D_T^{st} and $\mathcal{D}_T^{st}$ that yield new semantics for autoepistemic logic: the semantics of extensions, the semantics of partial extensions and the well-founded semantics.

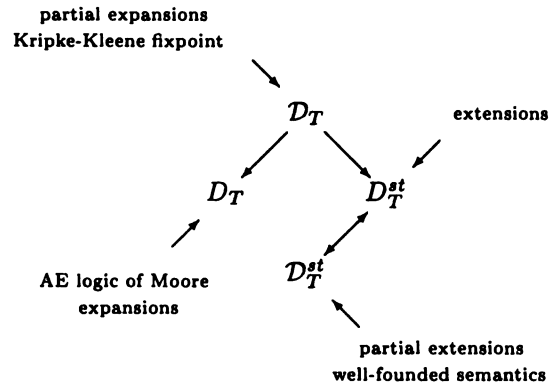


Figure 1: Operators associated with autoepistemic logic

4 DEFAULT LOGIC

While possible-world semantics played a prominent role in the study of autoepistemic logics [Moo84, Lev90, DMT98] they have not, up to now, had a similar impact on default logic. In this section we will introduce a comprehensive semantic treatment of default logic in terms of possible-world structures and belief pairs. Our approach will follow closely that used in the preceding sections.

We observed earlier that autoepistemic logic can be viewed as the logic of the operator $\mathcal{D}_T$. Its fixpoints, and fixpoints of the operators that can be derived from $\mathcal{D}_T$, determine all major semantics for autoepistemic logic. We will now develop a similar treatment of default logic.

As before, we start with a 2-valued truth function that gives a conservative estimate of the logical value of a formula or a default with respect to a belief pair (P, S) and an interpretation I. For a propositional formula φ, we define $\mathcal{H}^{dl}_{(P,S),I}(\varphi) = I(\varphi)$. For a default $d = \frac{\alpha:\beta_1,\ldots,\beta_k}{\gamma}$, we set $\mathcal{H}^{dl}_{(P,S),I}(d) = \mathbf{t}$ if at least one of the following conditions holds:

1. there is $J \in S$ such that $J(\alpha) = \mathbf{f}$
2. there is i, $1 \leq i \leq k$ such that for every $J \in P$, $J(\beta_i) = \mathbf{f}$
3. $I(\gamma) = \mathbf{t}$.

We set $\mathcal{H}^{dl}_{(P,S),I}(d) = \mathbf{f}$, otherwise. Clearly, the definition of $\mathcal{H}^{dl}_{(P,S),I}(d)$ agrees with the intuitive reading of a default d: it is true, according to a conservative point of view, if its prerequisite is false (even with respect to a liberal view captured by S) or if at least one of its justifications is definitely impossible (it is false according to a conservative point of view captured by P) or if its consequent is true (in I). As before, we can also argue that $\mathcal{H}^{dl}_{(S,P),I}(d)$ provides a liberal estimate for a truth value of d with respect to (P, S) (the roles of P and S are reversed).

Let $\Delta = (D, W)$ be a default theory. We use the truth function $\mathcal{H}^{dl}_{(P,S),I}$ to define an operator $\mathcal{E}_\Delta$ on

the lattice $\mathcal{B}$ of belief pairs:

$$\mathcal{E}_\Delta(P,S) = (\mathcal{E}^l_\Delta(P,S), \mathcal{E}^u_\Delta(P,S)),$$

where

$$\mathcal{E}^l_\Delta(P,S) = \{I : \mathcal{H}^{dl}_{(S,P),I}(\Delta) = \mathbf{t}\}$$

and

$$\mathcal{E}^u_\Delta(P,S) = \{I : \mathcal{H}^{dl}_{(P,S),I}(\Delta) = \mathbf{t}\}.$$

This definition can be justified similarly as that of the operator $\mathcal{D}_T$ in Section 3.

We will now define a 2-valued version of the operator $\mathcal{E}_\Delta$. To this end, we use the following result.

Theorem 4.1 *Let Δ be a default theory. If $B \in \mathcal{B}$ is complete then $\mathcal{E}_\Delta(B)$ is also complete.*

Let Q be a possible-world structure. We define

$$E_\Delta(Q) = Q',$$

where Q' is a possible-world structure such that $\mathcal{E}_\Delta(Q,Q) = (Q',Q')$ (its existence is guaranteed by Theorem 4.1).

To the best of our knowledge, the operator E_Δ has not appeared explicitly in the literature before. Its fixpoints, however, did. In [MT89a], the concept of a *weak extension* of a default theory was introduced and studied (the approach used there was proof-theoretic). It turns out that fixpoints of the operator E_Δ correspond precisely to weak extensions of Δ. Thus, the semantics given by the operator E_Δ is precisely the semantics of weak extensions.

Theorem 4.2 *Let Δ be a default theory. Then:*

1. *A propositional theory T is a weak extension of a default theory Δ according to* [MT89a] *if and only if $T = \{\varphi : I(\varphi) = \mathbf{t},$ for every $I \in Q\}$ for a fixpoint Q of E_Δ.*
2. *A possible-world structure Q is a fixpoint of E_Δ if and only if a belief pair (Q,Q) is a fixpoint of $\mathcal{E}_\Delta$.*

In view of Theorem 4.2(1), we call fixpoints of the operator E_Δ *weak extensions*. Theorem 4.2(2) implies that complete fixpoints of the operator $\mathcal{E}_\Delta$ are in one-to-one correspondence with the fixpoints of the operator E_Δ. Thus, we call fixpoints of the operator $\mathcal{E}_\Delta$ — *partial weak extensions* (they can be regarded as a 4-valued generalization of weak extensions, consistent fixpoints can be regarded as 3-valued generalizations).

The key property of the operator $\mathcal{E}_\Delta$ is its $\preceq_{kn}$-monotonicity. Thus, $\mathcal{E}_\Delta$ has a least fixpoint. We call it the *Kripke-Kleene fixpoint* and denote it by $KK(\Delta)$. We refer to the corresponding semantics as the *Kripke-Kleene semantics* for Δ.

The Kripke-Kleene semantics can be obtained by iterating the operator $\mathcal{E}_\Delta$ starting with the least informative belief pair $(\mathcal{A},\emptyset)$. Thus, it has a constructive flavor. Second, it approximates the skeptical reasoning with weak extensions and provides a test for uniqueness of a weak extension.

Theorem 4.3 *Let Δ be a default theory and let $KK(\Delta) = (P,S)$.*

1. *The fixpoint $KK(\Delta)$ is consistent, that is, $S \subseteq P$.*
2. *If $I(\varphi) = \mathbf{t}$ for every $I \in P$, then φ belongs to every weak extension. If $J(\varphi) = \mathbf{f}$ for some $J \in S$, φ does not belong to any weak extension.*
3. *If $P = S$ (that is, if $KK(\Delta)$ is complete) then Δ has a unique weak extension corresponding to the possible-world structure P.*

The Kripke-Kleene semantics is computationally attractive. Deciding whether a formula is in the intersection of the weak extensions of a default theory Δ is Π^2_P-complete. In contrast, by adapting the methods developed in [DMT98] to the case of default logic we can show that the problem of computing $KK(\Delta)$ (specifically, assuming $KK(\Delta) = (P,S)$, deciding whether $I(\varphi) = \mathbf{t}$ for every $I \in P$, or whether $J(\varphi) = \mathbf{f}$ for some $J \in S$) is in the class Δ^2_P.

So far we have not yet reconstructed the concept of an extension. In order to do so, we will now derive from $\mathcal{E}_\Delta$ two other operators related to default logic. Let us consider a belief pair (P,S). We want to revise it to a belief pair (P',S'). We might do it by fixing S and taking for P' a preferred revision of P, and by fixing P and taking for S' a preferred revision of S.

It is easy to see that $\preceq_{kn}$-monotonicity of $\mathcal{E}_\Delta$ implies that the operator $\mathcal{E}^l_\Delta(\cdot,S)$ is $\sqsubseteq$-monotone operator on $\mathcal{W}$. Consequently, it has a least fixpoint. This fixpoint can be taken as the preferred way to revise P given S. Thus, we define

$$E^{st}_\Delta(S) = lfp((\mathcal{E}_\Delta)^l(\cdot,S))$$

As in the case of autoepistemic logic, one can see that E^{st}_Δ also specifies the preferred way to revise S given P, that is $E^{st}_\Delta(P) = lfp(\mathcal{E}^u_\Delta(P,\cdot))$. Thus, we define the operator on $\mathcal{B}$ as follows:

$$\mathcal{E}^{st}_\Delta(P,S) = (E^{st}_\Delta(S), E^{st}_\Delta(P)).$$

It turns out that the concept of extension as defined by Reiter can be obtained from the operator E^{st}_{Δ}. It is known that Reiter's extensions are theories of fixpoints of the operator Σ_{Δ} introduced by Guerreiro and Casanova [GC90] (see also [Lif90, MT93]). One can show that the operator E^{st}_{Δ} coincides with the operator Σ_{Δ}. Thus, we have the following result.

Theorem 4.4 *A theory T is an extension of a default theory Δ if and only if $T = \{\varphi : I(\varphi) = \mathbf{t}$, for every $I \in Q\}$ for some fixpoint Q of E^{st}_{Δ}.*

In view of Theorem 4.4, we refer to the fixpoints of E^{st}_{Δ} as *extensions*. We have the following result relating fixpoints of the operators E^{st}_{Δ} and $\mathcal{E}^{st}_{\Delta}$.

Theorem 4.5 *Let Δ be a default theory. For every possible-world structure P, P is a fixpoint of E^{st}_{Δ} if and only if (P, P) is a fixpoint of $\mathcal{E}^{st}_{\Delta}$.*

It follows that the fixpoints of $\mathcal{E}^{st}_{\Delta}$ can be regarded as 4-valued (3-valued, in the case of consistent fixpoints) generalizations of an extension of a default theory. We will therefore call them *partial extensions*. It turns out that partial extensions coincide with stationary extensions defined in [PP94].

Our next result describes monotonicity properties of the operators E^{st}_{Δ} and $\mathcal{E}^{st}_{\Delta}$.

Theorem 4.6 *Let Δ be a default theory. Then, the operator E^{st}_{Δ} is $\sqsubseteq$-antimonotone and the operator $\mathcal{E}^{st}_{\Delta}$ is $\preceq_{kn}$-monotone.*

Theorem 4.6 implies that the operator $\mathcal{E}^{st}_{\Delta}$ has a least fixpoint. We will denote it by $WF(\Delta)$ and refer to it as the *well-founded fixpoint* of Δ. We will call the semantics it implies a *well-founded semantics* of Δ. The well-founded semantics of Δ coincides with the well-founded semantics of default logic introduced by Baral and Subrahmanian [BS91]. The well-founded semantics allows us to approximate skeptical reasoning with extensions and yields a sufficient condition for the uniqueness of an extension.

Theorem 4.7 *Let Δ be a default theory and let $WF(\Delta) = (P, S)$. Then:*

1. *The fixpoint $WF(\Delta)$ is consistent.*
2. *For every partial extensions B of $\mathcal{E}^{st}_{\Delta}$, $WF(\Delta) \preceq_{kn} B$.*
3. *If $J(\varphi) = \mathbf{t}$ for every $I \in P$, then φ belongs to every extension. If $J(\varphi) = \mathbf{f}$ for some $J \in S$, φ does not belong to any extension.*
4. *If $P = S$ (that is, if $WF(\Delta)$ is complete) then Δ has a unique extension corresponding to the possible-world structure P.*

Well-founded semantics has a constructive flavor. It can be obtained by iterating the operator $\mathcal{E}^{st}_{\Delta}$ over the belief pair $(\mathcal{A}, \emptyset)$. In addition, by extending the approach described in [DMT98], one can show that the problem of computing the well-founded semantics is in the class Δ^2_P.

Finally, let us note connections between (partial) weak extensions and (partial) extensions, and between the Kripke-Kleene and well-founded semantics for default logic.

Theorem 4.8 *Let Δ be a default theory. Then:*

1. *$KK(\Delta) \preceq_{kn} WF(\Delta)$.*
2. *Every extension of Δ is a $\sqsubseteq$-minimal weak extension of Δ.*
3. *Every partial extension (P, S) of Δ is a minimal partial weak extension of Δ in the following sense: for every partial weak extension (P', S'), if $P' \sqsubseteq P$ and $S' \sqsubseteq S$, then $P = P'$ and $S = S'$.*

In summary, default logic can be viewed as the logic of the operator $\mathcal{E}_{\Delta}$. Its fixpoints define the semantics of partial weak extensions. The least fixpoint of $\mathcal{E}_{\Delta}$ defines the Kripke-Kleene semantics. The operator $\mathcal{E}_{\Delta}$ gives rise to the operator E_{Δ}, which yields the semantics of weak extensions. Kripke-Kleene semantics provides an approximation for the skeptical reasoning under the semantics of weak extensions. The operator $\mathcal{E}_{\Delta}$ also leads to the operator $\mathcal{E}^{st}_{\Delta}$. Consistent fixpoints of this operator yield stationary extensions. Fixpoints of a related operator E^{st}_{Δ}, defined on the lattice $\mathcal{W}$, correspond to extensions by Reiter. The least fixpoint of the operator $\mathcal{E}^{st}_{\Delta}$ results in the well-founded semantics for default logic and approximates the skeptical reasoning under the semantics of extensions. The relationships between the operators of default logic are illustrated in Figure 2.

5 DEFAULT LOGIC VERSUS AUTOEPISTEMIC LOGIC

The results of the paper shed new light on the relationship between default and autoepistemic logics. The nature of this relationship was the subject of extensive investigations since the time both systems were introduced in early 80s. Konolige [Kon88] proposed to encode a default $d = \frac{\alpha : \beta_1, \ldots, \beta_k}{\gamma}$ by the modal formula

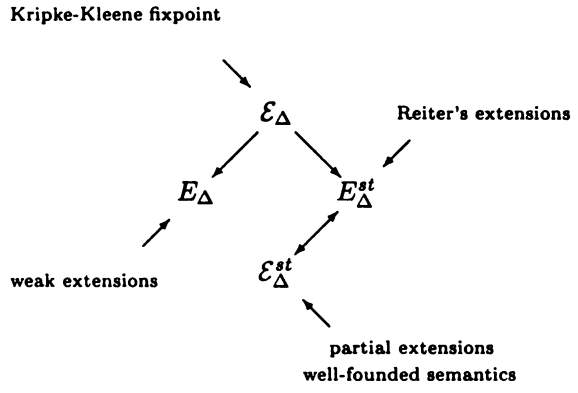


Figure 2: Operators associated with default logic

$m(d) = K\alpha \wedge \neg K\neg\beta_1 \wedge \ldots \neg K\neg\beta_k \supset \gamma$ and to represent a default theory $\Delta = (D, W)$ by a modal theory $m(\Delta) = W \cup \{m(d) : d \in D\}$. Despite the fact that the encoding is intuitive it does not provide a correspondence between default logic as defined by Reiter and autoepistemic logic as defined by Moore. Consider a default theory Δ with $W = \emptyset$ and $D = \{\frac{p:q}{p}\}$, where p and q are two different atoms. Then Δ has exactly one extension, $Cn(\emptyset)$. Applying the translation of Konolige to Δ yields the theory $m(\Delta) = \{Kp \wedge \neg K\neg q \supset p\}$. The theory $m(\Delta)$ has two expansions. One of them is generated by the theory $Cn(\emptyset)$ and corresponds to the only extension of Δ. The other expansion is generated by the theory $Cn(\{p\})$. Thus, the Konolige's translation does not give a one-to-one correspondence between extensions of default theories and expansions of their modal encodings.

This mismatch can be explained within the semantic framework introduced in the paper. Konolige's translation does not establish correspondence between extensions and expansions because they are associated with different operators. Expansions are associated with fixpoints of the operator D_T. Its counterpart on the side of default logic is the operator E_Δ. Fixpoints of this operator are not extensions but weak extensions of Δ. Extensions turn out to be associated with the operator E^{st}_Δ. Its counterpart on the side of autoepistemic logic is the operator D^{st}_T, introduced in Section 3. This operator, to the best of our knowledge, has not appeared in the literature and properties of its fixpoints and the relationship to McDermott-Doyle style logics [MD80, McD82, MT93] are not known.

Once we properly align concepts from default logic with those from autoepistemic logic, Konolige's translation works! This alignment is illustrated in Figure 3 and is formally described in the following theorem.

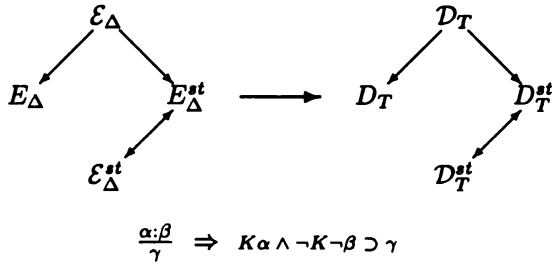


Figure 3: Embedding default logic into autoepistemic logic

Theorem 5.1 *Let Δ be a default theory and let $T = m(\Delta)$. Then the following pairs of operators coincide and, thus, have the same fixpoints:*

1. *$E_\Delta = D_T$ (that is, weak extensions correspond to expansions).*
2. *$\mathcal{E}_\Delta = \mathcal{D}_T$ (that is, partial weak extensions correspond to partial expansions; Kripke-Kleene semantics for Δ and Kripke-Kleene semantics for T coincide).*
3. *E^{st}_Δ and D^{st}_T (that is, extensions correspond to strong expansions).*
4. *$\mathcal{E}^{st}_\Delta$ and $\mathcal{D}^{st}_T$ (that is, partial extensions correspond to partial strong expansions, well-founded semantics for Δ and well-founded semantics for T coincide).*

6 DISCUSSION AND FUTURE WORK

We presented results uncovering the semantic properties of default and autoepistemic logics. In each case, a whole family of semantics can be derived from a single operator by purely algebraic transformations. Most importantly, the translation of Konolige establishes a perfect correspondence between the families of semantics of default and autoepistemic logics. This elegant picture can be further extended to the case of logic programming. As discovered by Fitting, all key semantics for logic programs can be similarly obtained from a single operator, the 4-valued van Emden-Kowalski one-step provability operator [vEK76]. The resulting semantic structure for logic programming is shown in Figure 4.

In addition, the translation of logic program clauses into default rules proposed in [BF91, MT89b] establishes an embedding of logic programming into default

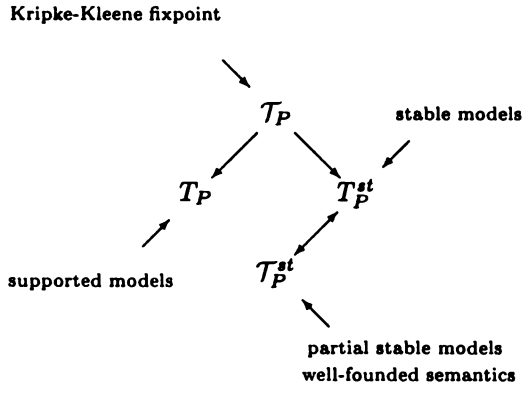


Figure 4: Operators associated with logic programming

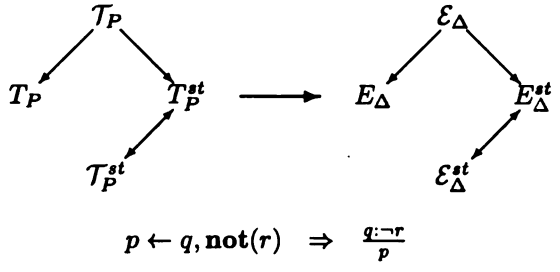


Figure 5: Embedding logic programming in default logic

logic that precisely aligns the corresponding semantics (Figure 5).

Let us further note that the approach to semantics of nonmonotonic logics presented here can also be extended to the case of reflexive autoepistemic logic by Schwarz [Sch95]. As in all other cases discussed in this paper, all major semantics for the reflexive autoepistemic logic can be obtained from a single approximating operator closely related to the operator $\mathcal{D}_T$.

Our work also points to an interesting open problem in the area of modal nonmonotonic logics. The fixpoints of the operator D_T^{st} have not, to the best of our knowledge, been studied in the literature. In particular, it is not known if they can be described in the McDermott-Doyle scheme [MD80, McD82] as $\mathcal{S}$-expansions for some modal logic $\mathcal{S}$.

Finally, let us mention that it is possible to develop an abstract, purely algebraic treatment of the concept of approximations. It generalizes the approach presented here and the work of Fitting on logic programming semantics. An account of this abstract treatment of approximations can be found in [DMT00].

Acknowledgements

This work was partially supported by the NSF grants CDA-9502645 and IRI-9619233.

References

[BF91] N. Bidoit and C. Froidevaux. Negation by default and unstratifiable logic programs. *Theoretical Computer Science*, 78(1, (Part B)):85–112, 1991.

[BS91] C. Baral and V.S. Subrahmanian. Dualities between alternative semantics for logic programming and nonmonotonic reasoning (extended abstract). In A. Nerode, W. Marek, and V.S. Subrahmanian, editors, *Logic programming and non-monotonic reasoning (Washington, DC, 1991)*, pages 69–86, Cambridge, MA, 1991. MIT Press.

[Che80] B.F. Chellas. *Modal logic. An introduction.* Cambridge University Press, Cambridge-New York, 1980.

[DMT98] M. Denecker, V. Marek, and M. Truszczyński. Fixpoint 3-valued semantics for autoepistemic logic. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 840 – 845. MIT Press, 1998.

[DMT00] M. Denecker, V. Marek, and M. Truszczyński. Unified semantic treatment of default and autoepistemic logics. In *Principles of Knowledge Representation and Reasoning, Proceedings of the Seventh International Conference (KR2000).* Morgan Kaufmann Publishers, 2000.

[Fit99] M. C. Fitting. Fixpoint semantics for logic programming – a survey. *Theoretical Computer Science*, 1999. To appear.

[GC90] R. Guerreiro and M. Casanova. An alternative semantics for default logic. Preprint. The Third International Workshop on Nonmonotonic Reasoning, South Lake Tahoe, 1990.

[Got95] G. Gottlob. Translating default logic into standard autoepistemic logic. *Journal of the ACM*, 42(4):711–740, 1995.

[Kon88] K. Konolige. On the relation between default and autoepistemic logic. *Artificial Intelligence*, 35(3):343–382, 1988.

[Lev90] H. J. Levesque. All I know: a study in autoepistemic logic. *Artificial Intelligence*, 42(2-3):263–309, 1990.

[Lif90] V. Lifschitz. On open defaults. In J. Lloyd, editor, *Proceedings of the symposium on computational logic*, pages 80–95. Berlin: Springer-Verlag, 1990. ESPRIT Basic Research Series.

[McD82] D. McDermott. Nonmonotonic logic II: nonmonotonic modal theories. *Journal of the ACM*, 29(1):33–57, 1982.

[MD80] D. McDermott and J. Doyle. Nonmonotonic logic I. *Artificial Intelligence*, 13(1-2):41–72, 1980.

[Moo84] R.C. Moore. Possible-world semantics for autoepistemic logic. In *Proceedings of the Workshop on Non-Monotonic Reasoning*, pages 344–354, 1984. Reprinted in: M. Ginsberg, ed., *Readings on nonmonotonic reasoning*, pp. 137–142, Morgan Kaufmann, 1990.

[MT89a] W. Marek and M. Truszczyński. Relating autoepistemic and default logics. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (Toronto, ON, 1989)*, Morgan Kaufmann Series in Representation and Reasoning, pages 276–288, San Mateo, CA, 1989. Morgan Kaufmann.

[MT89b] W. Marek and M. Truszczyński. Stable semantics for logic programs and default theories. In E.Lusk and R. Overbeek, editors, *Proceedings of the North American Conference on Logic Programming*, pages 243–256. MIT Press, 1989.

[MT93] W. Marek and M. Truszczyński. *Nonmonotonic logics; context-dependent reasoning*. Springer-Verlag, Berlin, 1993.

[PP94] H. Przymusińska and T. Przymusiński. Stationary default extensions. *Fundamenta Informaticae*, 21(1-2):67–87, 1994.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.

[Sch95] J. Schlipf. The expressive powers of the logic programming semantics. *Journal of the Computer Systems and Science*, 51(1):64–86, 1995.

[Tar55] A. Tarski. Lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[Tru91] M. Truszczyński. Modal interpretations of default logic. In *Proceedings of IJCAI-91*, pages 393–398, San Mateo, CA, 1991. Morgan Kaufmann.

[vEK76] M.H. van Emden and R.A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.

[VRS91] A. Van Gelder, K.A. Ross, and J.S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

Missionaries and Cannibals in the Causal Calculator

Vladimir Lifschitz
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712, USA

Abstract

A knowledge representation formalism is "elaboration tolerant" to the extent that it is convenient to modify a set of facts expressed in the formalism to take into account new phenomena or changed circumstances. John McCarthy illustrated this idea by defining 19 elaborations of the Missionaries and Cannibals Problem. We argue that, to a certain degree, the goal of elaboration tolerance is met by the input language of Norman McCain's Causal Calculator. We present formal descriptions of the basic Missionaries and Cannibals Problem and of ten of McCarthy's enhancements as input files accepted by the Causal Calculator. Each enhancement is obtained from the basic formulation by the simplest kind of elaboration—adding postulates.

1 Introduction

This note is about *elaboration tolerance*—a fundamental idea in knowledge representation, described by John McCarthy [9] as follows:

> A formalism is elaboration tolerant to the extent that it is convenient to modify a set of facts expressed in the formalism to take into account new phenomena or changed circumstances... Human-level AI will require representations with much more elaboration tolerance than those used by present AI programs, because human-level AI needs to be able to take new phenomena into account.
>
> The simplest kind of elaboration is the addition of new formulas. Next comes changing the values of parameters. Adding new arguments to functions and predicates represents more of a change.

McCarthy illustrates this idea by defining 19 elaborations of the Missionaries and Cannibals Problem (MCP): "Three missionaries and three cannibals come to a river and find a boat that holds two. If the cannibals ever outnumber the missionaries on either bank, the missionaries will be eaten. How shall they cross?" In one of his elaborations, the boat leaks and must be bailed concurrently with rowing; in others, an oar has to be recovered from the opposite bank of the river, or cannibals can be converted into missionaries, and so forth. The challenge is to design a knowledge representation formalism that would allow us to perform these elaborations by simple means, preferably by just adding new formulas.

This paper argues that, to a certain degree, this goal is met by the input language of the Causal Calculator (CCALC)—a system for query answering and satisfiability planning designed and implemented at the University of Texas.[1] Presented below are formal descriptions of the basic form of MCP and of (certain interpretations of) ten of its enhancements from [9] as input files accepted by CCALC. Each enhancement is obtained from the basic MCP by the simplest kind of elaboration—adding postulates. To stress this fact, the common part of all 11 forms of MCP is "factored out" and placed in a separate file, `common.t`, which is included in each of the input files using the `include` directive of CCALC. The input file for the basic version of MCP, `basic.t`, is very short: essentially the only thing it adds to `common.t` is the statement of the initial conditions and the goal. (The extension `.t` in the names of both files indicates that the files are intended for the use of CCALC in the "transition mode," discussed in the next section.)

[1] `http://www.cs.utexas.edu/users/tag/cc/` .

For each version except one, CCALC has found a solution and verified that the problem could not be solved in fewer steps. Our interest in automatic plan generation is orthogonal to the investigation of elaboration tolerance, which is a property of knowledge representation formalisms—not of algorithms or software systems. But, as a practical matter, it is convenient to have a planner available as a debugging tool.

The main ideas of this paper are not special to the input language of CCALC; other action languages and logic-based planners could be used instead. But CCALC provides a convenient platform for demonstrating these ideas.

After a brief discussion of the Causal Calculator in Section 2, we describe how two ideas related to elaboration tolerance—the use of the abnormality predicate [8] and describing objects in terms of attributes—apply to CCALC (Sections 3–5). In Section 6 a formalization of the basic MCP is presented, and Sections 7–16 describe its elaborations. In Section 17 we comment on some of the remaining challenges.

2 Transition Mode of the Causal Calculator

The original version of CCALC was written by Norman McCain as part of his dissertation [5]; later on, he was assisted by Esra Erdem, Vladimir Lifschitz, Norman Richards, Armando Tacchella and Hudson Turner. Some experimental results on CCALC planning are reported in [7].

In the "transition mode," CCALC operates with a description of a *transition system*—a directed graph whose edges correspond to the transitions caused by the execution of actions. The standard include file `C.t` allows the user to describe transition systems in action language $\mathcal{C}$ from [1].

Here is an example:

```
% File 'example.t'

:- include 'C.t'.

:- sorts
 vessel; loc.

:- variables
 V                  :: vessel;
 L,L1               :: loc.

:- constants
 boat               :: vessel;
 b1, b2             :: loc;
 at(vessel,loc)     :: inertialTrueFluent;
 cross(vessel,loc)  :: action.

 cross(V,L) causes at(V,L).
 nonexecutable cross(V,L) if at(V,L).

 always \/ L: at(V,L).
 caused -at(V,L1) if at(V,L) && -(L=L1).

:- plan
facts ::
 0: at(boat,b1);
goals ::
 1: at(boat,b2).
```

This file describes a domain consisting of objects of two kinds: vessels and locations; `V` is a variable for vessels, and `L`, `L1` are variables for locations. There exists one vessel `boat`, and there are two locations—banks `b1`, `b2`.

A state of the transition system defined in this file is characterized by the truth values of the fluents `at(boat,b1)` and `at(boat,b2)`. As discussed below, in each state of the system one of these fluents is true and the other is false, so that only two states are possible: S_1 (`boat` is at `b1`) and S_2 (`boat` is at `b2`).

In this transition system, an action is a subset of the elementary actions `cross(boat,b1)`, `cross(boat,b2)`. The execution of an action is understood as the concurrent execution of the elementary actions that belong to it. However, the first action (crossing to `b1`) is not executable in state S_1, and the second is not executable in state S_2, so that, in this example, the concurrent execution of more than one elementary action is never possible. The empty action—waiting—can be performed in either state.

The sorts `inertialTrueFluent` and `action` are defined in `C.t`. The use of `inertialTrueFluent` rather than `fluent` in the declaration of `at` tells CCALC that this fluent, once true, tends to remain true. The solution to the frame problem incorporated in CCALC is based on the ideas of [10], [11] and [6].

The constant declarations are followed by four propositions. The first two describe the effect of `cross(V,L)` (it causes `at(V,L)` to become true) and its precondition (it is nonexecutable if `at(V,L)` is true). The `always` proposition asserts that, in every state, every vessel is at a certain location (`\/` is the CCALC sign for the existential quantifier, or rather for the corresponding finite disjunction). This proposition tells us that the fluents `at(boat,b1)` and `at(boat,b2)` cannot be

simultaneously false. The fourth proposition says that if a vessel `V` is at a certain location then there is a cause for `V` not to be at any other location (in CCALC, - denotes negation and `&&` denotes conjunction). This proposition tells us that the fluents `at(boat,b1)` and `at(boat,b2)` cannot be simultaneously true; in addition, it makes `-at(V,L1)` an indirect effect of any action that causes `at(V,L)` with `L` different from `L1`.

The last part of the file describes a planning problem: given that `at(boat,b1)` is true at time 0, make `at(boat,b2)` true at time 1.

Given this input file, CCALC replaces each of the given propositions with its ground instances, compiles the grounded propositions into a set of propositional formulas similar to those used in satisfiability planning [3], and calls a propositional solver (SATO [12] is the default) to find a solution. It produces the output

```
calling sato...
run time (seconds)                0.00

0.  at(boat,b1)

ACTIONS: cross(boat,b2)

1.  at(boat,b2)
```

that shows a one-step plan, along with the fluents that are true at each of the time instants 0, 1 when that plan is executed.

3 Abnormality

In the interests of elaboration tolerance, it is useful to make some propositions of CCALC defeasible. Imagine, for instance, that we want to enhance the boat domain description above by the actions of breaking a vessel into pieces and shipping the pieces to other locations. In such an enhancement, the assertion that every vessel has a location

```
always \/ L: at(V,L)
```

would no longer be valid. An elaboration tolerant formalism should allow us to retract such an assertion, in the spirit of nonmonotonic reasoning, by adding new postulates.

To make a constraint defeasible, we will "label" it, for instance:

```
(l1(V)) always \/ L: at(V,L).
```

One of the macro definitions in the standard file `C.t` expands this line into

```
always -ab(l1(V)) ->> \/ L: at(V,L).
```

(`->>` is material implication). We understand `always` to mean "in all states"; a labeled `always` means "in all states, normally" or "in all states, by default." The symbol `ab` is declared in `C.t` as follows:

```
:- constants
 ab(abLabel) :: defaultFalseFluent.
```

A label has to be declared before it is used:

```
:- constants
 l1(vessel) :: abLabel.
```

To retract the existence of location constraint when the vessel is broken into pieces we would add a proposition like this:

```
caused ab(l1(V)) if broken(V).
```

In the context of MCP, the labeling mechanism is needed, for instance, to formalize McCarthy's Elaboration 17: "if the strongest of the missionaries rows fast enough, the cannibals won't have gotten so hungry that they will eat the missionaries." The assumption that missionaries are never outnumbered by cannibals in file `common.t` is labeled, and then it is partially retracted in file `jmc17.t`.

In our formalizations of MCP, abnormality labels are used with many `always`, `never` and `nonexecutable` propositions. Greater flexibility can be achieved by including many or all free variables of a proposition in its abnormality label as arguments. But here, for simplicity, we use abnormality labels without arguments.

Introducing abnormality labels required some changes in standard file `C.t`, but not in the code of CCALC.

4 Action Attributes

From the perspective of MCP, representing the action of crossing the river by an expression like `cross(boat,b2)` is inadequate: this expression does not provide enough information to decide how crossing affects the numbers of missionaries and cannibals on both banks (unless boarding the boat is treated as a separate action). An expression like `cross(boat,b2,1,1)` (1 missionary and 1 cannibal use the boat to move to Bank 2) would do.

Some of McCarthy's elaborations would require, however, that `cross` be given even more additional arguments. In Elaboration 17 mentioned above, we distinguish between rowing fast and rowing slowly; that would require expressions like

`cross(boat,b2,1,1,fast)`. In Elaboration 6, only one missionary and one cannibal can row. In this case, the expression denoting the action should tell us which of the people crossing can row.

As discussed in the introduction, adding arguments to functions and predicates is what we want to avoid: our goal is to perform all elaborations by adding postulates. To achieve this goal, we will think of an action as something that can be described more completely or less completely, depending on how many details about its execution are provided. The action of crossing the river will be denoted by the symbol `cross` without arguments. To say that a vessel `V` is used to cross, we will write `cross_in(V)`. To say that the destination is `L`, we will write `cross_to(L)`. Syntactically, `cross_in(V)` and `cross_to(L)` are action symbols, like `cross`. But we will think of them as merely "attributes," or details of execution, of action `cross`. This intuition will be expressed by the postulates asserting that

- if the action `cross` is executed then there exists a unique `V` such that `cross_in(V)` holds;
- if the action is not executed then `cross_in(V)` does not hold for any `V`;

similarly,

- if the action `cross` is executed then there exists a unique `L` such that `cross_to(L)` holds;
- if the action is not executed then `cross_to(L)` does not hold for any `L`.

New elaborations will involve extending the language by new attribute symbols, instead of adding new arguments to the existing action symbols. The advantage of this method is that the description of the effects of an action given earlier does not have to be modified when an enhancement is introduced. Instead, we add causal laws that describe new effects of the action in terms of its newly introduced attributes.

Technically, using action attributes is similar to "operator splitting" [4, 2]—the representation method that uses expressions like

$$Source(a, 3) \wedge Object(b, 3) \wedge Destination(c, 3)$$

to say that b was moved from a to c at time 3, instead of $Move(a, b, c, 3)$. Operator splitting was invented to speed up search. Our use of attributes is motivated by elaboration tolerance.

Here is the example from Section 3 rewritten in terms of attributes:

```
% File 'common1.t': action of crossing
% and two of its attributes

:- include 'C.t'.       % standard file

:- macros
 attribute -> action.

:- sorts
 vessel; loc.

:- variables
 V,V1                  :: vessel;
 L,L1                  :: loc.

:- constants
 at(vessel,loc)        :: inertialTrueFluent;
 cross                 :: action;
 cross_in(vessel)      :: attribute;
 cross_to(loc)         :: attribute.

% cross_in and cross_to are attributes
% of action cross
nonexecutable -(cross <-> \/ V: cross_in(V)).
nonexecutable -(cross <-> \/ L: cross_to(L)).
nonexecutable cross_in(V) && cross_in(V1)
                                    if -(V=V1).
nonexecutable cross_to(L) && cross_to(L1)
                                    if -(L=L1).

% Properties of crossing
cross_in(V) && cross_to(L) causes at(V,L).
nonexecutable cross_in(V) && cross_to(L)
                                 if at(V,L).

% Properties of at
always \/ L: at(V,L).
caused -at(V,L1) if at(V,L) && -(L=L1).
```

This file does not define any constants and does not contain the planning problem found at the end of `example.t`. The following file includes `common1.t` and contains the two missing components:

```
% 'common1-test.t'

:- include 'common1.t'.

:- constants
 boat                  :: vessel;
 b1, b2                :: loc.

:- plan
facts ::
 0: at(boat,b1);
```

```
goals ::
 1: at(boat,b2).
```

Given `common1-test.t` as input, CCALC produces the following output:

```
calling sato...
run time (seconds)                0.01

0.  at(boat,b1)

ACTIONS: cross cross_in(boat) cross_to(b2)

1.  at(boat,b2)
```

We understand the list

```
cross cross_in(boat) cross_to(b2)
```

in the output of CCALC as follows: The action to be executed is crossing the river; the river is to be crossed in the boat; the destination is the second bank. Future versions of CCALC will possibly format plans expressed in terms of attributes differently, perhaps like this:

```
cross(in:boat, to:b2).
```

They may also allow us express postulates like

```
nonexecutable -(cross <-> \/ V: cross_in(V)).
nonexecutable -(cross <-> \/ L: cross_to(L)).
nonexecutable cross_in(V) && cross_in(V1)
                                  if -(V=V1).
nonexecutable cross_to(L) && cross_to(L1)
                                  if -(L=L1).
```

more concisely—by saying that `in` and `to` are attributes of `cross`.

5 Groups Crossing

To illustrate the use of attributes for achieving the goal of elaboration tolerance, we will enhance `common1.t` by new attributes. For every "group" (such as missionaries or cannibals) we can ask how many members of the group are aboard while the action `cross` is executed. This number is the attribute of `cross` corresponding to the given group. The counting mechanism introduced in this elaboration will be used for many purposes—to count oars in Elaboration 5, to count those who can row in Elaboration 6, and so forth.

```
% 'common2.t': extending 'common1.t' by
% new attributes of crossing

:- include 'common1.t'.

:- sorts
 number; group.

:- variables
 M,N                              :: number;
 G                                :: group;
 X                                :: computed.

:- constants
 0..maxInt                      :: number;
 num(group,loc,number) :: inertialTrueFluent;
 cross_howmany(group,number) :: attribute.

% cross_howmany is a family of attributes
nonexecutable
    -(cross <-> \/ N: cross_howmany(G,N)).
nonexecutable
    cross_howmany(G,M) && cross_howmany(G,N)
    if -(M=N).

% addition and subtraction in the
% range 0..maxInt
:- macros
 sum(#1,#2,#3) ->
        #1 is min((#2)+(#3),maxInt);
 diff(#1,#2,#3) ->
        #1 is max((#2)-(#3),0).

% properties of crossing described in
% terms of the new attributes
cross_in(V) && cross_howmany(G,M)
   causes num(G,L,X)
   if at(V,L) && num(G,L,N) && diff(X,N,M).
cross_in(V) && cross_to(L)
            && cross_howmany(G,M)
              causes num(G,L,X)
              if num(G,L,N) && sum(X,N,M).
nonexecutable
   cross_in(V) && cross_howmany(G,M)
   if at(V,L) && num(G,L,N) && M>N.

% properties of num
always \/ N: num(G,L,N).
caused -num(G,L,N) if num(G,L,M) && -(M=N).
```

(By declaring `X` to be a `computed` variable we tell CCALC that, in the process of grounding, there will be no need to select values for `X`; they will have been always pre-computed.)

The propositions describing the additional effects of action `cross` say that if the action is performed by `M` members of a group then the number of members of the group at the departure location decreases by `M`, and

the number of members of the group at destination increases by M. This assertion is correct only if cross is not executed concurrently with any other action that affects these two numbers. This is an essential limitation of the representation used in common2.t. The basic MCP and most of its elaborations introduced in [9] do not allow such "difficult" concurrency (most of them involve no concurrency at all), but there are exceptions. We will return to this question several times.

To test the new theory of crossing, we'll assume that there are 3 people initially on one bank and 2 on the other, and ask CCALC to move everyone to the second bank in one step:

```
% 'common2-test.t'

:- macros
 maxInt -> 6.

:- include 'common2.t'.

:- constants
 boat              :: vessel;
 b1, b2            :: loc;
 person            :: group.

:- plan
facts ::
 0: at(boat,b1),
 0: num(person,b1,3),
 0: num(person,b2,2);
goals ::
 1: num(person,b1,0).
```

The output:

```
calling sato...
run time (seconds)                    0.02

0.  at(boat,b1)  num(person,b1,3)
num(person,b2,2)

ACTIONS: cross cross_in(boat)
cross_to(b2) cross_howmany(person,3)

1.  at(boat,b2)  num(person,b1,0)
num(person,b2,5)
```

6 Basic MCP

Now we are ready to show file common.t mentioned in the introduction—the file included in all 11 versions of MCP.

```
% 'common.t': common for all forms of MCP

:- include 'common2.t'.

:- variables
 K                          :: number.

:- constants
 boat                       :: vessel;
 b1, b2                     :: loc;
 mi,ca                      :: group;
 capacity(vessel,number)    :: exogenousFluent.

:- macros
% #1 missionaries are outnumbered
% by #2 cannibals
 outnumbered(#1,#2)
      -> (#2 > #1) && (#1 > 0).

% labels to be used for reference if we
% decide to retract propositions below
:- constants
  1..5 :: abLabel.

% missionaries should not be outnumbered
% in any location
(1) never num(mi,L,M) && num(ca,L,N)
                      && outnumbered(M,N).

% additional preconditions for crossing

% someone should be in the boat
(2) nonexecutable cross_howmany(mi,0)
                  && cross_howmany(ca,0).

% but not too many
(3) nonexecutable
     cross_in(V) && cross_howmany(mi,M)
                 && cross_howmany(ca,N)
   if capacity(V,K) && sum(X,M,N) && X>K.

% missionaries should not be outnumbered
% on the way
(4) nonexecutable cross_howmany(mi,M)
                  && cross_howmany(ca,N)
    if outnumbered(M,N).

% boat capacity
always
    capacity(V,M) && capacity(V,N) ->> M=N.
(5) always capacity(boat,2).
```

The basic form of MCP can be formalized as follows:

```
% 'basic.t': original MCP
```

```
:- macros
 maxInt -> 3.

:- include 'common.t'.

:- plan
facts ::
 0: (num(mi,b1,3) && num(ca,b1,3)),
 0: (num(mi,b2,0) && num(ca,b2,0)),
 0: at(boat,b1);
goals ::
 10..11: (num(mi,b1,0) && num(ca,b1,0)).
```

The expression 10..11 in the last line instructs CCALC to try to find a plan of length 10 and then, if there is no such plan, try length 11. Since the length of the shortest plan for the basic form of MCP is actually 11, CCALC finds such a plan after determining that a shorter plan is impossible. The output begins with

```
calling sato...
  No plan of length 10,
run time (seconds)                6.84
calling sato...
run time (seconds)               17.59
```

which is followed by the usual 11 step solution.

7 Boat Can Carry Three

The boat can carry three. Then four pairs can cross.

The assumption in common.t that the boat can only carry two has an abnormality label. This fact allows us to retract it:

```
% 'jmc4.t': McCarthy's Elaboration 4

:- macros
 maxInt -> 4.

:- include 'common.t'.

% retract (5) from 'common.t'

caused ab(5).

:- constants
 6 :: abLabel.

(6) always capacity(boat,3).

:- plan
facts ::
 0: (num(mi,b1,4) && num(ca,b1,4)),
 0: (num(mi,b2,0) && num(ca,b2,0)),
 0: at(boat,b1);
goals ::
 8..9: (num(mi,b1,0) && num(ca,b1,0)).
```

SATO run times: 7 and 18 seconds.

8 An Oar on each Bank

"There is an oar on each bank. One person can cross in the boat with just one oar, but two oars are needed if the boat is to carry two people. We can send a cannibal to get the oar and then we are reduced to the original problem" [9].

We'll count oars using the group mechanism used above to count missionaries and cannibals.

```
% 'jmc5.t': McCarthy's Elaboration 5

:- macros
 maxInt -> 3.

:- include 'common.t'.

:- constants
 oars       :: group;
 6,7        :: abLabel.

(6) nonexecutable
  cross_in(boat) && cross_howmany(oars,0).
(7) nonexecutable
 cross_in(boat) && cross_howmany(oars,1) &&
 cross_howmany(mi,M) && cross_howmany(ca,N)
    if M+N > 1.

:- plan
facts ::
 0: (num(mi,b1,3) && num(ca,b1,3)
     && num(oars,b1,1)),
 0: (num(mi,b2,0) && num(ca,b2,0)
     && num(oars,b2,1)),
 0: at(boat,b1);
goals ::
 12..13: (num(mi,b1,0) && num(ca,b1,0)).
```

SATO run times: 27 and 44 seconds.

9 Not Everyone Can Row

Only one missionary and one cannibal can row. The problem is still solvable, but now the length of the shortest plan is 13.

In addition to the group mi of missionaries, we in-

troduce its subset `rowers(mi)`, and similarly for cannibals. To avoid the possibility of iterating function `rowers` (which would create an infinite set of ground terms), we limit it to "basic groups."

The fact that any basic group `BG` is a superset of `rowers(BG)` is reflected in the postulates that prohibit the states in which the number of members of `BG` in any location is smaller than the number of members of `rowers(BG)` in the same location, as well as the actions in which the number of members of `BG` crossing the river is smaller than the number of members of `rowers(BG)` crossing.

```
% 'jmc6.t':  McCarthy's Elaboration 6

:- macros
 maxInt -> 3.

:- include 'common.t'.

:- sorts
 group >> basicGroup.

:- variables
 BG                    :: basicGroup.

:- constants
 mi,ca                 :: basicGroup;
 rowers(basicGroup)    :: group.

never
 num(rowers(BG),L,M) && num(BG,L,N) && M>N.

nonexecutable cross_howmany(rowers(BG),M)
             && cross_howmany(BG,N) if M>N.

:- constants
 6 :: abLabel.

(6) nonexecutable
             cross_howmany(rowers(mi),0)
             && cross_howmany(rowers(ca),0).

:- plan
facts ::
 0: (num(mi,b1,3) && num(ca,b1,3)),
 0: (num(rowers(mi),b1,1)
             && num(rowers(ca),b1,1)),
 0: (num(mi,b2,0) && num(ca,b2,0)),
 0: at(boat,b1);
goals ::
 12..13: (num(mi,b1,0) && num(ca,b1,0)).
```

SATO run times: 286 and 273 seconds.

10 A Very Big Cannibal

The biggest cannibal cannot fit in the boat with another person. The problem is still solvable, but now the length of the shortest plan turns out to be 15.

We can keep track of the position of the biggest cannibal using the group mechanism introduced in Section 5 if we treat him as a one-person group.

```
% 'jmc8.t':  McCarthy's Elaboration 8

:- macros
 maxInt -> 3.

:- include 'common.t'.

:- constants
 vbc :: group. % very big cannibal

never num(vbc,L,M) && num(ca,L,N) && M>N.

nonexecutable cross_howmany(vbc,M)
             && cross_howmany(ca,N) && M>N.

:- constants
 6 :: abLabel.

(6) nonexecutable cross_howmany(vbc,1)
          && cross_howmany(mi,M)
          && cross_howmany(ca,N) if M+N>1.

:- plan
facts ::
 0: (num(mi,b1,3) && num(ca,b1,3)
                  && num(vbc,b1,1)),
 0: (num(mi,b2,0) && num(ca,b2,0)
                  && num(vbc,b2,0)),
 0: at(boat,b1);
goals ::
 14..15: (num(mi,b1,0) && num(ca,b1,0)).
```

SATO run times: 2149 and 9746 seconds.

11 Big Cannibal and Small Missionary

If the biggest cannibal is isolated with the smallest missionary—in the boat or on either bank—the latter will be eaten. A plan that works for the basic MCP can be adapted to this enhancement if we specify, for every crossing, whether the biggest cannibal and the smallest missionary are aboard.

```
% 'jmc9.t':  McCarthy's Elaboration 9
:- macros
```

```
 maxInt -> 3.

:- include 'common.t'.

:- constants
 bca :: group;        % big cannibal
 smi :: group.        % small missionary

never num(bca,L,M) && num(ca,L,N) && M>N.

never num(smi,L,M) && num(mi,L,N) && M>N.

nonexecutable cross_howmany(bca,M)
&& cross_howmany(ca,N) && M>N.

nonexecutable cross_howmany(smi,M)
        && cross_howmany(mi,N) && M>N.

:- constants
 6,7 :: abLabel.

(6) never num(bca,L,1) && num(smi,L,1)
                       && num(mi,L,1).
(7) nonexecutable cross_howmany(bca,1)
              && cross_howmany(smi,1).

:- plan
facts ::
 0: (num(mi,b1,3) && num(ca,b1,3)
 && num(bca,b1,1) && num(smi,b1,1)),
 0: (num(mi,b2,0) && num(ca,b2,0)),
 0: at(boat,b1);
goals ::
 10..11: (num(mi,b1,0) && num(ca,b1,0)).
```

SATO run times: 18 and 22 seconds.

12 Converting Cannibals

Three missionaries alone with a cannibal can convert him into a missionary. We'll treat converting as an action that can be included in the plan, rather than an event that happens whenever the preconditions are satisfied.

Do we allow crossing the river and converting a cannibal to occur in parallel? Can a solution begin, for instance, with two cannibals crossing to Bank 2 while the third cannibal is being converted into a missionary on Bank 1? If yes, this is an example of "difficult" concurrency (Section 5) that the approach of this paper does not allow.

Let's assume that the concurrent execution of the two actions involved is not allowed in a plan. Even so, the problem can be solved in 10 steps, including one conversion, instead of 11 steps required for the basic MCP.

```
% 'jmc11.t': McCarthy's Elaboration 11

:- macros
 maxInt -> 6.

:- include 'common.t'.

:- constants
 convert          :: action;
 convert_at(loc)  :: attribute.

% convert_at is an attribute of convert
nonexecutable -(convert
            <-> \/ L: convert_at(L)).
nonexecutable convert_at(L) && convert_at(L1)
            if -(L=L1).
convert_at(L) causes num(mi,L,X)
            if num(mi,L,N) && sum(X,N,1).
convert_at(L) causes num(ca,L,X)
            if num(ca,L,N) && diff(X,N,1).

:- constants
 6,7 :: abLabel.

(6) nonexecutable convert_at(L)
            if num(mi,L,N) && N<3.
(7) nonexecutable convert_at(L)
            if num(ca,L,N) && -(N=1).

% no concurrent actions
nonexecutable cross && convert.

:- plan
facts ::
 0: (num(mi,b1,3) && num(ca,b1,3)),
 0: (num(mi,b2,0) && num(ca,b2,0)),
 0: at(boat,b1);
goals ::
 9..10: (num(mi,b1,0) && num(ca,b1,0)).
```

SATO run times: 37 and 55 seconds.

13 The Bridge

"There is a bridge. This makes it obvious to a person that any number can cross provided two people can cross at once... There is no need to get rid of the boat unless this is a part of the elaboration wanted" [9]. In the formalization below we do not get rid of the boat, but using the bridge and the boat concurrently

is prohibited to avoid "difficult" concurrency.

```
% 'jmc13.t':  McCarthy's Elaboration 13

:- macros
 maxInt -> 5.

:- include 'common.t'.

:- constants
 useBridge                  :: action;
 useBridge_from(loc)        :: attribute;
 useBridge_to(loc)          :: attribute;
 useBridge_howmany(group,number)
                            :: attribute.

% useBridge_from, useBridge_to and
% useBridge_howmany are attributes:

 nonexecutable -(useBridge
        <-> \/ L: useBridge_from(L)).
 nonexecutable -(useBridge
        <-> \/ L: useBridge_to(L)).
 nonexecutable -(useBridge
        <-> \/ N: useBridge_howmany(G,N)).
 nonexecutable useBridge_from(L)
        && useBridge_from(L1) if -(L=L1).
 nonexecutable useBridge_to(L)
        && useBridge_to(L1) if -(L=L1).
 nonexecutable useBridge_howmany(G,M)
        && useBridge_howmany(G,N) if -(M=N).

% properties of using bridge
useBridge_from(L) && useBridge_howmany(G,M)
        causes num(G,L,X)
        if num(G,L,N) && diff(X,N,M).
useBridge_to(L) && useBridge_howmany(G,M)
        causes num(G,L,X)
        if num(G,L,N) && sum(X,N,M).
nonexecutable useBridge_from(L)
        && useBridge_howmany(G,M)
        if num(G,L,N) && M>N.
nonexecutable useBridge_from(L)
        && useBridge_to(L1)
        if -((L=b1 && L1=b2)
             ++ (L=b2 && L1=b1)).

:- constants
 6 :: abLabel.

(6) nonexecutable useBridge_howmany(mi,M)
     && useBridge_howmany(ca,N) if M+N > 2.

% no concurrent actions
nonexecutable cross && useBridge.

:- plan
facts ::
 0: (num(mi,b1,5) && num(ca,b1,5)),
 0: (num(mi,b2,0) && num(ca,b2,0)),
 0: at(boat,b1);
goals ::
 4..5: (num(mi,b1,0) && num(ca,b1,0)).
```

SATO run times: 2 seconds each time.

14 The Boat Leaks

The boat leaks and must be bailed concurrently with rowing. To make the problem more interesting, we assume that bailing is only needed when two persons are crossing, and that one person would not be able to row and to bail simultaneously.

In this example concurrency is essential, but it does not present the problem discussed in Section 5: bailing does not change the numbers of missionaries and cannibals in any location.

```
% 'jmc14.t': McCarthy's Elaboration 14

:- macros
 maxInt -> 3.

:- include 'common.t'.

:- constants
 bail       :: action;
 6,7        :: abLabel.

(6) nonexecutable cross_howmany(mi,M)
    && cross_howmany(ca,N) && -bail
                  if sum(X,M,N) && X>1.
(7) nonexecutable cross_howmany(mi,M)
    && cross_howmany(ca,N) && bail
                  if sum(X,M,N) && X=1.

:- plan
facts ::
 0: (num(mi,b1,3) && num(ca,b1,3)),
 0: (num(mi,b2,0) && num(ca,b2,0)),
 0: at(boat,b1);
goals ::
 10..11: (num(mi,b1,0) && num(ca,b1,0)).
```

SATO run times: 7 and 9 seconds.

15 The Island

There is an island. Then any number can cross.

Formalizing this enhancement is straightforward: we just need to declare `island` to be an additional location. But computationally the problem becomes too difficult for CCALC (or rather for SATO), even if we assume only 4 missionaries and 4 cannibals. To make the computational task easier, we modify the goal as follows: one missionary should stay on Bank 1, one cannibal should be on the island, and the rest of the company, along with the boat, should reach Bank 2. Once this subgoal is achieved, the original goal can be achieved by sending a cannibal to bring the missing missionary and then the missing cannibal to Bank 2 (4 moves).

The output of CCALC shows that the modified planning problem can be solved in 11 steps. Consequently, the original problem can be solved in 15 steps.

```
% 'jmc16.t':  McCarthy's Elaboration 16

:- macros
 maxInt -> 4.

:- include 'common.t'.

:- constants
 island :: loc.

:- plan
facts ::
 0: (num(mi,b1,4) && num(ca,b1,4)),
 0: (num(mi,island,0) && num(ca,island,0)),
 0: (num(mi,b2,0) && num(ca,b2,0)),
 0: at(boat,b1);
goals ::
 10..11: (num(mi,b1,1) && num(ca,b1,0)
     && num(mi,island,0) && num(ca,island,1)
     && at(boat,b2)).
```

SATO run times: 192 and 1894 seconds.

16 Cannibals Are Not Hungry

"There are four cannibals and four missionaries, but if the strongest of the missionaries rows fast enough, the cannibals won't have gotten so hungry that they will eat the missionaries. This could be made precise in various ways, but the information is usable even in vague form" [9].

We assume that rowing fast is done so fast that, even when executed many times, it doesn't give cannibals enough time to get hungry; rowing slowly is so slow that cannibals get hungry after you do it just once.

```
% 'jmc17.t':  McCarthy's Elaboration 17

:- macros
 maxInt -> 4.

:- include 'common.t'.

:- sorts
 speed.

:- variables
 S, S1              :: speed.

:- constants
 slowly, fast   :: speed;
 strmi          :: group;
 hungry         :: defaultTrueFluent;
 cross_howfast(speed) :: attribute.

% cross_howfast is an attribute
nonexecutable -(cross <->
      \/ S: cross_howfast(S)).
nonexecutable cross_howfast(S)
      && cross_howfast(S1) if -(S=S1).
never num(strmi,L,M) && num(mi,L,N) && M>N.
nonexecutable cross_howmany(strmi,M) &&
       cross_howmany(mi,N) && M>N.

:- constants
 6 :: abLabel.

(6) nonexecutable cross_howfast(fast)
                 && cross_howmany(strmi,0).

% retract (1) from 'common.t'
caused ab(1) if -hungry.

% retract (4) from 'common.t'
caused ab(4) if -hungry.

cross_howfast(fast) causes -hungry
                    if -hungry.

:- plan
facts ::
 0: -hungry,
 0: (num(mi,b1,4) && num(strmi,b1,1)
     && num(ca,b1,4)),
 0: (num(mi,b2,0) && num(ca,b2,0)),
 0: at(boat,b1);
goals ::
```

```
12..13: (num(mi,b1,0) && num(ca,b1,0)).
```

SATO run times: 1006 and 7361 seconds.

17 Conclusion

In this paper we argue that the input language of CCALC provides a partial solution to the problem of elaboration tolerance in representing actions. Our examples use two innovations in the methodology of representing knowledge in CCALC—abnormality labels and describing actions in terms of their attributes.

We have presented here formalizations of 10 out of the 19 elaborations of MCP described in [9]. Some of the remaining nine present interesting topics for future work.

One significant limitation of our approach to MCP is mentioned in Section 5: our description of the additional effects of crossing on the numbers of missionaries and cannibals at various locations presupposes that actions affecting these numbers are not executed in parallel. This limitation prevented us from allowing concurrency in the discusion of Elaboration 11 (converting cannibals) and Elaboration 13 (the bridge). It has also prevented us from formalizing McCarthy's Elaboration 10: One of the missionaries is Jesus Christ; four can cross. Walking on water will help only if performed concurrently with crossing by boat, and this is an example of "difficult" concurrency. It should be stressed that this difficulty is not in any way inherent in the language of CCALC; see the section on interaction between concurrent actions in [1].

In Elaboration 19 there are two sets of missionaries and cannibals, each with one boat, too far apart along the river to interact. To allow the two boats to be used concurrently, we would need to modify the representation of the `cross` action in terms of attributes used in this note: `cross` would have to be given an argument.

Challenges of different kinds are found in Elaboration 12 (the use of probabilities) and Elaboration 15 (splitting an event into parts).

Acknowledgements

A representation of MCP related to the work described above has been investigated by Mary Heidt. Esra Erdem, Michael Gelfond, Joohyung Lee, John McCarthy, Emilio Remolina and Hudson Turner provided useful comments on earlier drafts of this note. Norman McCain helped with the design of abnormality labels. Thanks to all of them for help and for sharing their ideas with me.

This work was partially supported by National Science Foundation under grant IIS-9732744.

References

[1] Enrico Giunchiglia and Vladimir Lifschitz. An action language based on causal explanation: Preliminary report. In *Proc. AAAI-98*, pages 623–630, 1998.

[2] Henry Kautz, David McAllester, and Bart Selman. Encoding plans in propositional logic. In *Principles of Knowledge Representation and Reasoning: Proc. of the Fifth Int'l Conf.*, pages 374–384, 1996.

[3] Henry Kautz and Bart Selman. Planning as satisfiability. In *Proc. ECAI-92*, pages 359–363, 1992.

[4] Henry Kautz and Bart Selman. Pushing the envelope: planning, propositional logic and stochastic search. In *Proc. AAAI-96*, pages 1194–1201, 1996.

[5] Norman McCain. *Causality in Commonsense Reasoning about Actions.*[2] PhD thesis, University of Texas at Austin, 1997.

[6] Norman McCain and Hudson Turner. Causal theories of action and change. In *Proc. AAAI-97*, pages 460–465, 1997.

[7] Norman McCain and Hudson Turner. Satisfiability planning with causal theories. In Anthony Cohn, Lenhart Schubert, and Stuart Shapiro, editors, *Proc. Sixth Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 212–223, 1998.

[8] John McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 26(3):89–116, 1986.

[9] John McCarthy. Elaboration tolerance.[3] In progress, 1999.

[10] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[11] Hudson Turner. Representing actions in logic programs and default theories: a situation calculus approach. *Journal of Logic Programming*, 31:245–298, 1997.

[12] Hantao Zhang. An efficient propositional prover. In *Proc. CADE-97*, 1997.

[2] `ftp://ftp.cs.utexas.edu/pub/techreports/tr97-25.ps.Z` .

[3] `http://www-formal.stanford.edu/jmc/elaboration.html` .

Representation of Action

Narratives as Programs

Ray Reiter
Department of Computer Science
University of Toronto
Toronto, Canada
M5S 3G4
reiter@cs.toronto.edu

Abstract

Representing narratives, and reasoning about them, has been a prominent theme in logical formalisms for dynamical systems (e.g. [9, 5, 2]). However, the existing literature provides a rather limited concept of what a narrative is; the examples all concern linear action sequences, sometimes including incomplete information about action occurrence times. The point of departure of this paper is the observation that narratives often include more complex constructions, including nondeterminism, loops, and recursive procedures. Therefore, we propose that, in their full generality, narratives are best viewed as *programs*. In many cases, the situation calculus-based programming language GOLOG is suitable for this purpose. In this setting, we define what it means to query a narrative, and discover that this task is formally identical to proving the correctness of programs as studied in computer science. Since this general task is hopelessly intractable, we focus on procedure-free narrative programs and for a wide class of such programs and queries we prove that a regression-based approach to query evaluation is correct. Finally, we describe a Prolog implementation of these ideas.

1 Introduction

Current logical theories of actions can be divided into two broad categories: *narrative-based* approaches (e.g. features and fluents [17], the event calculus and its relatives [18, 6, 1]) that rely on linear temporal logics, and *situation-based* approaches (e.g. the families of action languages [3], various dialects of the situation calculus [8, 13, 9]) that provide for branching futures. As Reiter [15] observed, these two classes of action theories require quite different reasoning mechanisms in their treatment of planning problems: abduction for the former, deduction for the latter. Similarly, they differ in how they represent and reason about narratives – stories about action occurrences, including possibly their temporal properties. Narrative-based approaches treat narratives in a clean and straightforward way; reasoning consists of logical deduction using a knowledge base of assertions about a narrative's action occurrences and their temporal relations. The situation-based approaches suffer in comparison, at least judging by the existing literature; they require more elaborate mechanisms, usually extending their basic ontology to include the concept of an action occurrence, perhaps also together with an "actual path" in the tree of histories (e.g. [10, 11]; but see [9, 12] for different viewpoints). For a rich account of both narratives and hypothetical reasoning, which also requires an enriched ontology, see [5].

Regardless of the approach, the existing literature provides a rather limited concept of what a narrative is; the examples all concern linear action sequences, sometimes including incomplete information about action occurrence times. In this paper, we propose a much richer notion; specifically, we suggest that, in their full generality, narratives are best viewed as nondeterministic programs, and we investigate the suitability of the situation calculus-based programming language GOLOG for this purpose. In this setting, we define what it means to query a narrative, and discover that this task is formally identical to proving the correctness of programs as studied in computer science. Since this general task is hopelessly intractable, we focus on procedure-free narrative programs and for a wide class of such programs and queries we prove that a regression-based approach to query evaluation is correct. Finally, we describe a Prolog implementation of these ideas.

One consequence of the above narratives-as-GOLOG-programs perspective will be a cleaner account of narratives for situation-based theories of action than those of [10, 11]. Another will be a more general notion of narrative that can perhaps be profitably incorporated into narrative-based theories of actions as well.

2 Narratives as GOLOG Programs

The central intuition we wish to convey about narratives is that, in their most general forms, they are *programs*. Consider the following example stories about a blocks world, and their representations as programs:

1. Pat moved block A onto B, after which he moved C onto the table.

   $move(A, B)\,;\, moveToTable(C)$.[1]

   Here, ; stands for the standard sequence operator of conventional programming languages.

2. First, Pat moved a block onto the table, then she moved A onto it.

   $(\pi\, x).moveToTable(x)\,;\, move(A, x)$.

   Here, $(\pi\, x)$ is a nondeterministic operator that chooses an arbitrary argument of an action expression. So the above program means: nondeterministically choose an x, and for that x, move it to the table, then move A onto it.

3. First, Pat moved block A onto block B or block C – I don't know which – then he moved D onto some block on the table.

   $[move(A, B) \mid move(A, C)]\,;$
   $(\pi\, x)[onTable(x)?\,;\, move(D, x)]$.

   Here we have introduced two new operators: $A_1 \mid A_2$ means nondeterministically choose one of the actions A_1, A_2 to perform, and $\phi?$ means test the truth value of the logical expression ϕ.

4. Pat cleared all the blocks from the table by putting them on the floor.

   while $(\exists x)onTable(x)$ **do**
   $(\pi\, x).onTable(x)?\,;\, moveToFloor(x)$.

5. Pat unstacked all the towers onto the table.

   $unstackTowers$.

   Here, $unstackTowers$ is a recursive procedure:

   proc $unstackTowers$
   $(\forall x)onTable(x)? \mid$
   $(\pi\, x)moveToTable(x)\,;\, unstackTowers$
   endproc

6. Pat moved B to the table at time 10, and at some time before that, she moved a block onto A.

   $(\pi\, t)[t < 10?\,;\, (\pi\, x)move(x, A, t)]\,;$
   $moveToTable(B, 10)$.

   Here we have added a time parameter to action terms, as suggested for the situation calculus in [15]. We have cheated a bit in representing this narrative by recognizing that the second action mentioned in the natural language version actually preceded the first action mentioned. We take it as a separate (and nontrivial) problem to mechanically translate a natural language narrative into a program, especially getting the temporal precedences right, and we do not address that issue here.

7. Pat moved B onto A; then she observed that A is on some block on the table; then she moved C to B.

   $move(B, A)\,;\, [(\exists x)onTable(x) \wedge on(A, x)]?\,;$
   $move(C, B)$.

   Here we treat observational actions as test actions. Similarly, we represent narrative descriptions – It was a dark and stormy night when Pat first moved a block to the table – as test actions:

   $[dark \wedge stormy]?\,;\, (\pi\, x)moveToTable(x)$.

   The reason for this initially odd looking decision will become clear after we formalize a suitable programming language for representing narratives, and after we define the notion of querying a narrative.

The purpose of these examples is to argue that narratives are more complicated things that simple sequences of actions; they can have all the complexity of programs, including nondeterminism, loops and recursive procedures. But what kinds of programs can serve to represent narratives, and what can it mean to query a narrative viewed as a program? Our proposal will be to take the situation calculus-based programming language GOLOG as a suitable representation for narratives, and to appeal to GOLOG's semantics for the purposes of querying such narratives. We begin with a brief description of GOLOG.

2.1 GOLOG

GOLOG [7] is a language for defining complex actions in terms of a repertoire of primitive actions axiomatized in the situation calculus. It provides the standard – and some not so standard – control structures found in most Algol-like languages. We have just seen a number of examples of GOLOG programs and control structures, together with their intuitive semantics.

GOLOG's formal semantics is specified by introducing an abbreviation $Do(\delta, s, s')$, where δ is a program and s and s' are situation terms. $Do(\delta, s, s')$ is best viewed as a macro that expands into a second order situation calculus formula; moreover, *that formula says that sit-*

[1] To simplify the notation, we suppress the actors (here, Pat) in our action terms.

uation s' can be reached from situation s by executing some sequence of actions specified by δ. Note that our programs may be nondeterministic, that is, may have multiple executions terminating in different situations.

Do is defined inductively on the structure of its first argument as follows:

1. *Primitive actions:* If α is a primitive action term,
$$Do(\alpha, s, s') \stackrel{def}{=} Poss(\alpha, s) \wedge s' = do(\alpha, s).$$
2. *Test actions:* When ϕ is a situation-suppressed logical expression,
$$Do(\phi?, s, s') \stackrel{def}{=} \phi[s] \wedge s = s'.$$
Here, $\phi[s]$ denotes the result of restoring the situation argument s to all of the situation-suppressed fluents mentioned by ϕ.
3. *Sequence:*
$$Do(\delta_1; \delta_2, s, s') \stackrel{def}{=} (\exists s^*).Do(\delta_1, s, s^*) \wedge Do(\delta_2, s^*, s').$$
4. *Nondeterministic choice of two actions:*
$$Do(\delta_1 \mid \delta_2, s, s') \stackrel{def}{=} Do(\delta_1, s, s') \vee Do(\delta_2, s, s').$$
5. *Nondeterministic choice of action arguments:*
$$Do((\pi x)\, \delta, s, s') \stackrel{def}{=} (\exists x)\, Do(\delta, s, s').$$
6. *Procedure calls:* For a predicate variable P whose last two arguments are the only ones of sort *situation*:
$$Do(P(t_1, \ldots, t_n), s, s') \stackrel{def}{=} P(t_1, \ldots, t_n, s, s').$$
7. *Blocks with local procedure declarations:*
$$Do(\{\textbf{proc}\ P_1(\vec{v}_1)\, \delta_1\ \textbf{endProc}\ ; \cdots ; \textbf{proc}\ P_n(\vec{v}_n)\, \delta_n\ \textbf{endProc}\ ;\ \delta_0\}, s, s') \stackrel{def}{=}$$
$$(\forall P_1, \ldots, P_n).[\bigwedge_{i=1}^{n} (\forall s_1, s_2, \vec{v}_i).Do(\delta_i, s_1, s_2) \supset P_i(\vec{v}_i, s_1, s_2)] \supset Do(\delta_0, s, s').$$

Other control structures, e.g. conditionals and while loops, can be defined in terms of the above constructs:

$$\textbf{if}\ test\ \textbf{then}\ prog_1\ \textbf{else}\ prog_2 \stackrel{def}{=} test?\ ;\ prog_1 \mid \neg test?\ ;\ prog_2$$

To define **while** loops, first introduce a nondeterministic iteration operator *, where $program^*$ means do *program* 0 or more times:

$$program^* \stackrel{def}{=} \textbf{proc}\ P()\ true? \mid [program;\ P()]\ \textbf{endProc};\ P()$$

Then **while** loops can be defined in terms of the * operator:

$$\textbf{while}\ test\ \textbf{do}\ program\ \textbf{endWhile} \stackrel{def}{=} [test?\ ;\ program]^*\ ;\ \neg test?$$

GOLOG programs are evaluated relative to a background theory of actions specifying a particular application domain. Specifically, if $\mathcal{D}$ is such a background action theory, and δ is a GOLOG program, then the evaluation of δ relative to $\mathcal{D}$ is defined to be the task of establishing the following entailment:

$$\mathcal{D} \models (\exists s) Do(\delta, S_0, s).$$

Any binding for the existentially quantified variable s obtained as a side effect of such a proof constitutes an execution trace of δ.

Example 2.1 Let ν be the narrative of example 3:

$$\nu = [move(A, B) \mid move(A, C)]\ ;\ (\pi x)[onTable(x)?\ ;\ (move(D, x)]$$

Then,

$$\begin{aligned} &Do(\nu, S_0, s) = \\ &\quad (\exists s').[Poss(move(A, B), S_0) \wedge \\ &\qquad s' = do(move(A, B), S_0) \vee \\ &\qquad Poss(move(A, C), S_0) \wedge \\ &\qquad s' = do(move(A, C), S_0)] \wedge \\ &\qquad (\exists x, s'').onTable(x, s') \wedge s' = s'' \wedge \\ &\qquad\quad Poss(move(D, x), s'') \wedge \\ &\qquad\quad s = do(move(D, x), s''). \end{aligned}$$

This is logically equivalent to the following formula, which is an example of a normal form to which $Do(\nu, S_0, s)$ can often be cast. Such normal forms will play an important role below, when we consider how to implement a system for querying narratives.

$$\begin{aligned} &Do(\nu, S_0, s) \equiv \\ &\quad (\exists x).Poss(move(A, B), S_0) \wedge \\ &\qquad onTable(x, do(move(A, B), S_0)) \wedge \\ &\qquad Poss(move(D, x), do(move(A, B), S_0)) \wedge \\ &\qquad s = do(move(D, x), do(move(A, B), S_0)) \\ &\quad \vee \\ &\quad (\exists x).Poss(move(A, C), S_0) \wedge \\ &\qquad onTable(x, do(move(A, C), S_0)) \wedge \\ &\qquad Poss(move(D, x), do(move(A, C), S_0)) \wedge \\ &\qquad s = do(move(D, x), do(move(A, C), S_0)). \end{aligned}$$

2.2 Reasoning about Narratives

A narrative is a description of action occurrences in some world. We take it that the central reasoning task for a narrative is to determine what the resulting world would be like. This task has three major components.

1. We need to infer the effects on the world of the action occurrences. This includes the actions' non-effects, so the frame problem must be taken into account. We therefore need a background, domain specific theory that characterizes action effects, and that solves the frame problem.
2. Observational actions and descriptions, as in Example 7 above, provide additional information about the world, and this must be taken into account in answering queries about a narrative.
3. Narratives convey implicit information about how the world must have been, by virtue of an action occurrence. For example, if a narrative claims that $move(A, B)$ was performed, then at the point that this action was performed, the action $move(A, B)$ must have been possible. So we can infer that, however else the world might have been, at the point of this action occurrence, both A and B must have been clear. Such information needs to be extracted from the narrative and made explicit for the purposes of answering queries about the narrative. Therefore, as part of the background theory, we require axioms specifying the action preconditions.

Finally, it is important to note that a convention about narratives is that one cannot assume more about the way the world is than is conveyed by the narrative and its immediate context. So, for example, when faced with example 1 above, one cannot, without further contextual justification, assume that A, B and C are all and only the available blocks. Neither can one assume particular initial locations for these blocks, for example, that A is initially on the table. In other words, narratives describe *open worlds*; one cannot assume complete information about the initial situation. Among other things, this observation precludes simple STRIPS-like action representations for describing and reasoning about narratives.

Definition 2.1 Query. A *query* is any situation calculus formula $Q(s)$ whose only free variable is the situation variable s.

Definition 2.2 Querying a Narrative
Let $Q(s)$ be a query, $\mathcal{D}$ a set of situation calculus axioms specifying an underlying domain of application, and ν a narrative viewed as a GOLOG program. Then the *answer to Q for the narrative ν relative to $\mathcal{D}$* is "yes" iff

$$\mathcal{D} \models (\forall s).Do(\nu, S_0, s) \supset Q(s).$$

The answer to Q is "no" iff the answer to $\neg Q$ is "yes".

Notice especially that in this form, querying a narrative is formally identical to the problem of proving properties of programs, as normally understood in computer science. We are simply asking whether all terminating situations s of the program ν have property Q. This is not good news for automating query evaluation for narratives; proving properties of programs is notoriously difficult, requiring mathematical induction for programs with loops and recursion, and it is unlikely that a general computational account can be given for this problem. For this reason, we shall limit ourselves in what follows to procedure-free programs, for the purposes of providing an implementation for this class of narratives.

3 Implementation Foundations

Here we provide the theoretical foundations for implementing and querying narratives when the underlying axioms form a basic action theory, and the GOLOG narrative program is procedure-free.

3.1 Basic Action Theories

Our concern in this paper will be with axioms for actions and their effects with a particular syntactic form ([13]).

Definition 3.1 Basic Action Theories
A *basic action theory* has the form

$\mathcal{D} = \Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$, where:

- Σ are the *foundational axioms for situations* [13]. These play no role in this paper and we omit them.
- $\mathcal{D}_{ap}$ is a set of *action precondition axioms*, one for each action function $A(\vec{x})$, of the form

$$Poss(A(\vec{x}, s) \equiv \Pi_A(\vec{x}, s). \quad (1)$$

  Here, $\Pi_A(\vec{x}, s)$ is a formula whose free variables are among $\vec{x}, s$, it does not quantify over situations, nor does it mention the predicate symbol $Poss$, and the only term of sort *situation* that it mentions is s.
- $\mathcal{D}_{ss}$ is a set of *successor state axioms*, one for each fluent $F(\vec{x}, s)$. These have the syntactic form

$$F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s), \quad (2)$$

  where $\Phi_F(\vec{x}, a, s)$ is a formula, all of whose free variables are among $a, s, \vec{x}$, it does not quantify over situations, nor does it mention the predicate symbol $Poss$, and the only term of sort *situation*

that it mentions is s. Such axioms embody a solution to the frame problem for deterministic actions [14].[2]

- $\mathcal{D}_{una}$ is the set of *unique names axioms* for all action function symbols.
- $\mathcal{D}_{S_0}$, the *initial database*, is a set of first order sentences such that no sentence of $\mathcal{D}_{S_0}$ quantifies over situations, or mentions *Poss*, and S_0 is the only term of sort *situation* mentioned by these sentences. $\mathcal{D}_{S_0}$ will function as the initial theory of the world.

Example 3.1 The following are the successor state and action precondition axioms for a blocks world basic action theory used in the implementation described below.

Action Precondition Axioms

$$Poss(move(x,y),s) \equiv clear(x,s) \wedge clear(y,s) \wedge x \neq y,$$

$$Poss(moveToTable(x),s) \equiv clear(x,s) \wedge \neg onTable(x,s).$$

Successor State Axioms

$$clear(x,do(a,s)) \equiv (\exists y)\{[(\exists z)a = move(y,z) \vee a = moveToTable(y)] \wedge on(y,x,s)\} \vee clear(x,s) \wedge \neg(\exists y)a = move(y,x),$$

$$on(x,y,do(a,s)) \equiv a = move(x,y) \vee on(x,y,s) \wedge a \neq moveToTable(x) \wedge \neg(\exists z)a = move(x,z),$$

$$onTable(x,do(a,s)) \equiv a = moveToTable(x) \vee onTable(x,s) \wedge \neg(\exists y)a = move(x,y).$$

Unique Names Axioms for Actions

$$move(x,y) \neq moveToTable(z),$$

$$move(x,y) = move(x',y') \supset x = x' \wedge y = y',$$

$$moveToTable(x) = moveToTable(x') \supset x = x'.$$

3.2 Regression for Basic Action Theories

Regression [13] is perhaps the single most important theorem-proving mechanism for the situation calculus; it provides a systematic way to establish that a basic action theory entails a so-called *regressable sentence*.

Definition 3.2 Concrete Situation Terms. These are inductively defined by: S_0 is a concrete situation term; when σ is a concrete situation term and α is an action term, then $do(\alpha,\sigma)$ is a concrete situation term.

[2] One can also give successor state axioms for functional fluents; because of space limitations, we do not discuss these here.

Definition 3.3 The Regressable Formulas. A regressable formula of the situation calculus is a first order formula W with the property that every situation term mentioned by W is concrete, and moreover, for every atom of the form $Poss(\alpha,\sigma)$ mentioned by W, α has the form $A(t_1,\ldots,t_n)$ for some n-ary action function symbol A and terms $t_1,\ldots,t_n$.

The essence of a regressable formula is that each of its *situation* terms is rooted at S_0, and therefore, one can tell, by inspection of such a term, exactly how many actions it involves. It is not necessary to be able to tell what those actions are, just how many they are. In addition, when a regressable formula mentions a *Poss* atom, we can tell, by inspection of that atom, exactly what is the action function symbol occurring in its first argument position, for example, that it is a *move* action.

The intuition underlying regression is this: Suppose we want to prove that a regressable sentence W is entailed by a basic action theory. Suppose further that W mentions a relational fluent atom $F(\vec{t},do(\alpha,\sigma))$, where F's successor state axiom is (2). By substituting $\Phi_F(\vec{t},\alpha,\sigma)$ for $F(\vec{t},do(\alpha,\sigma))$ in W we obtain a logically equivalent sentence W'. After we do so, the fluent atom $F(\vec{t},do(\alpha,\sigma))$, involving the complex situation term $do(\alpha,\sigma)$, has been eliminated from W in favour of $\Phi_F(\vec{t},\alpha,\sigma)$, and this involves the simpler situation term σ. In this sense, W' is "closer" to the initial situation S_0 than was W. Similarly, if W mentions an atom of the form $Poss(A(t_1,\ldots,t_n),\sigma)$,[3] there will be an action precondition axiom for A of the form (1) so we can eliminate this atom by replacing it with $\Pi_A(t_1,\cdots,t_n,\sigma)$. This process of replacing *Poss* and fluent atoms by the right hand sides of their action precondition and successor state axioms can be repeated until the resulting goal formula mentions only the situation term S_0. Regression is a mechanism that repeatedly performs the above reduction to a goal W, ultimately obtaining a logically equivalent goal W_0 whose only situation term is S_0. See [13] for precise definitions, and for a proof of the soundness and completeness of regression for basic action theories.

3.3 Procedure-Free Narratives

Definition 3.4 Choice Prenex Form
A GOLOG program is in *choice prenex form* iff it has the form

$$(\pi\,\vec{x}_1)P_1 \mid \cdots \mid (\pi\,\vec{x}_m)P_m,$$

where each P_i is of the form $\alpha_{i_1}\,;\,\cdots\,;\,\alpha_{i_k}$, $k \geq 1$, each

[3] Because W is assumed to be regressable, all *Poss* atoms mentioned by W must be of this form.

α is a primitive action or a test action, and association of the sequence operators in P_i is to the right.

Definition 3.5 Equivalence of Programs
Two GOLOG programs P_1 and P_2 are *equivalent* iff the following sentence is valid:

$$(\forall s, s').Do(P_1, s, s') \equiv Do(P_2, s, s').$$

Definition 3.6 Procedure-Free Programs
These are GOLOG programs defined without the procedure mechanism, i.e. using only primitive actions, tests (?), sequence (;) and nondeterministic choice (π and $|$).

Lemma 3.1 *There is an effective procedure for transforming a procedure-free GOLOG program into an equivalent program in choice prenex form.*

Proof: By repeatedly applying the following equivalence preserving transformations, until no further reductions are possible:[4]

$$\begin{array}{l} P_1 \,;\, (P_2 \mid P_3) \rightarrow P_1 \,;\, P_2 \mid P_1 \,;\, P_3, \\ (P_1 \mid P_2) \,;\, P_3 \rightarrow P_1 \,;\, P_3 \mid P_2 \,;\, P_3, \\ (P_1 \,;\, P_2) \,;\, P_3 \rightarrow P_1 \,;\, (P_2 \,;\, P_3), \\ (\pi x)[P_1 \mid P_2] \rightarrow (\pi x)P_1 \mid (\pi x)P_2, \\ [(\pi x)P_1] \,;\, P_2 \rightarrow (\pi x')[P_1|^x_{x'} \,;\, P_2], \\ P_1 \,;\, [(\pi x)P_2] \rightarrow (\pi x')[P_1 \,;\, P_2|^x_{x'}]. \end{array}$$

Here, x' is a new variable, distinct from any variable mentioned in P_1 or P_2. $P|^x_{x'}$ denotes the result of substituting x' for all free occurrences of x in P.
■

For programs δ that are specifically in choice prenex form, define a ternary relation $DoCpf(\delta, s, s')$ as follows:

1. *Primitive actions:* If α is a primitive action term,

$$DoCpf(\alpha, s, s') \stackrel{def}{=} Poss(\alpha, s) \wedge s' = do(\alpha, s).$$

$$DoCpf(\alpha \,;\, \delta, s, s') \stackrel{def}{=} Poss(\alpha, s) \wedge DoCpf(\delta, do(\alpha, s), s').$$

2. *Test actions:* When ϕ is a situation-suppressed logical expression,

$$DoCpf(\phi?, s, s') \stackrel{def}{=} \phi[s] \wedge s' = s.$$

$$DoCpf(\phi? \,;\, \delta, s, s') \stackrel{def}{=} \phi[s] \wedge DoCpf(\delta, s, s').$$

3. *Nondeterministic choice of two actions:*

$$DoCpf(\delta_1 \mid \delta_2, s, s') \stackrel{def}{=} DoCpf(\delta_1, s, s') \vee DoCpf(\delta_2, s, s').$$

4. *Nondeterministic choice of action arguments:*

$$DoCpf((\pi x)\,\delta, s, s') \stackrel{def}{=} (\exists x)\, DoCpf(\delta, s, s').$$

Lemma 3.2 *Let ν be a procedure-free GOLOG program that does not mention a choice operator (πa) where a is a variable of sort* action.[5] *Let κ be a choice prenex form for ν. Then*

1. $(\forall s).Do(\nu, S_0, s) \equiv DoCpf(\kappa, S_0, s)$ *is valid.*
2. $DoCpf(\kappa, S_0, s)$ *has the syntactic form:*

$$(\exists \vec{x}_1)[C_1 \wedge s = \sigma_1] \vee \cdots \vee (\exists \vec{x}_n)[C_n \wedge s = \sigma_n], \quad (3)$$

where each σ_i is a concrete situation term and each C_i is a regressable formula.

Proof: By Lemma 3.1, ν is equivalent to κ, and therefore, $(\forall s).Do(\nu, S_0, s) \equiv Do(\kappa, S_0, s)$ is valid.

Suppose

$$\kappa = (\pi \vec{x}_1)P_1 \mid \cdots \mid (\pi \vec{x}_m)P_m.$$

Then by the expansion rules for GOLOG programs of Section 2.1,

$$Do(\kappa, S_0, s) = (\exists \vec{x}_1)Do(P_1, S_0, s) \vee \cdots \vee (\exists \vec{x}_m)Do(P_m, S_0, s).$$

Similarly, by the rules for expanding $DoCpf$,

$$DoCpf(\kappa, S_0, s) = (\exists \vec{x}_1)DoCpf(P_1, S_0, s) \vee \cdots \vee (\exists \vec{x}_m)DoCpf(P_m, S_0, s).$$

Accordingly, it is sufficient to prove the following:

Suppose that a program P has the form $\alpha_1 \,;\, \cdots \,;\, \alpha_k$, where each α is a primitive action term or a test action, where the sequence operator associates to the right, and where P mentions free variables $\vec{x}$, none of which is of sort action. *Suppose further that S is a concrete situation term, possibly with free variables among $\vec{x}$. Then $(\forall \vec{x}, s).Do(P, S, s) \equiv DoCpf(P, S, s)$ is valid. Moreover, $DoCpf(P, S, s)$ is a formula of the form $C \wedge s = \sigma$, where σ is a concrete situation term, and C is a regressable formula.*

We prove this by induction on k.

Base case: P is just α, where α is a test action $\phi?$ or α is a primitive action term. In the first case,

$$Do(\alpha, S, s) = \phi[S] \wedge s = S = DoCpf(\alpha, S, s).$$

Now S is a concrete situation term. Moreover, because no free variable of P is an action variable, if

[4] Strictly speaking, we should prove that these are equivalence preserving transformations. The proofs are straightforward, and we omit the details.

[5] Of course, the program may mention a choice operator (πx) where x ranges over domain objects other than actions and situations, as in the examples of Section 2.

$\phi[S]$ mentions an atom of the form $Poss(\beta, S)$, then β cannot be an action variable, and therefore must be of the form $A(t_1, \ldots, t_n)$ for some n-ary action function symbol A and terms $t_1, \ldots, t_n$. Therefore, $\phi[S]$ is regressable, and we have established the base case when α is a test action.

Next, we establish the base case when α is a primitive action. As before, because no free variable of P is an action variable, α must be of the form $A(t_1, \ldots, t_n)$ for some n-ary action function symbol A and terms $t_1, \ldots, t_n$. Moreover,

$$\begin{aligned} Do(\alpha, S, s) &= Poss(\alpha, S) \wedge s = do(\alpha, S) \\ &= DoCpf(\alpha, S, s). \end{aligned}$$

Since α is of the form $A(t_1, \ldots, t_n)$, and since S is concrete, $Poss(\alpha, S)$ is regressable and $do(\alpha, S)$ is concrete. This establishes the remaining base case.

Induction step: Assume the result for $\alpha_1 ; \cdots ; \alpha_k$, $k \geq 1$, and suppose that α is a primitive action term. Then

$$\begin{aligned} &Do(\alpha ; \alpha_1 ; \cdots ; \alpha_k, S, s) = \\ &(\exists s^*)[Poss(\alpha, S) \wedge s^* = do(\alpha, S) \wedge \\ &\qquad Do(\alpha_1 ; \cdots ; \alpha_k, s^*, s)] \equiv \\ &Poss(\alpha, S) \wedge Do(\alpha_1 ; \cdots ; \alpha_k, do(\alpha, S), s). \end{aligned}$$

Now $do(\alpha, S)$ is concrete. Moreover, by the same argument as in the base case, $Poss(\alpha, S)$ is regressable. Finally, by induction hypothesis,

$$\begin{aligned} &Do(\alpha_1 ; \cdots ; \alpha_k, do(\alpha, S), s) \equiv \\ &\qquad DoCpf(\alpha_1 ; \cdots ; \alpha_k, do(\alpha, S), s), \end{aligned}$$

and $DoCpf(\alpha_1 ; \cdots ; \alpha_k, d(\alpha, S), s)$ is a formula of the form $C \wedge s = \sigma$, where C is regressible and σ is concrete. The result now follows.
The case where α is a test action is similar.
■

Example 3.2 With reference to Example 2.1, the last formula displayed is the one promised by the lemma.

Theorem 3.1 *Suppose that $\mathcal{D}$ is a basic action theory and that the program ν satisfies the conditions of Lemma 3.2. Suppose further that $Q(s)$ is a query with the property that $Q(\sigma)$ is regressable whenever σ is a concrete situation term. Without loss of generality, assume that the bound variables (if any) of $Q(s)$ are distinct from all of the variables $\vec{x}_i$ of (3). Then, with reference to Lemma 3.2,*

$$\mathcal{D} \models (\forall s).Do(\nu, S_0, s) \supset Q(s) \text{ iff for } i = 1, \ldots, n$$
$$\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models (\forall \vec{x}_i).\mathcal{R}[C_i] \supset \mathcal{R}[Q(\sigma_i)].$$

Moreover, $Q(\sigma_i)$ is a regressable formula. Here, $\mathcal{R}$ is the regression operator.

Proof: Let κ be a choice prenex form for ν. Then
$\mathcal{D} \models (\forall s).Do(\nu, S_0, s) \supset Q(s)$
iff, by Lemma 3.2,
$\mathcal{D} \models (\forall s).DoCpf(\kappa, S_0, s) \supset Q(s)$
iff, again by Lemma 3.2,

$$\begin{aligned} &\mathcal{D} \models \\ &\quad (\forall s).(\exists \vec{x}_1)[C_1 \wedge s = \sigma_1] \vee \cdots \vee (\exists \vec{x}_n)[C_n \wedge s = \sigma_n] \\ &\qquad \supset Q(s) \end{aligned}$$

iff, for $i = 1, \ldots, n$,

$$\mathcal{D} \models (\forall s).(\exists \vec{x}_i)[C_i \wedge s = \sigma_i] \supset Q(s)$$

iff, because the bound variables of Q are different then those of $\vec{x}_i$, and by properties of equality in first order logic,

$$\mathcal{D} \models (\forall \vec{x}_i).C_i \supset Q(\sigma_i).$$

By hypothesis, $Q(\sigma_i)$ is regressable because σ_i is concrete, and C_i is regressable by Lemma 3.2. Therefore, by the Regression Theorem of Pirri and Reiter [13],

$$\mathcal{D} \models (\forall \vec{x}_i).C_i \supset Q(\sigma_i)$$

iff

$$\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models (\forall \vec{x}_i).\mathcal{R}[C_i] \supset \mathcal{R}[Q(\sigma_i)].$$

■

This is our central computational result. Under suitable conditions on the program and query, narrative query evaluation can be done using regression; moreover, after performing the regression steps, all theorem proving is *relative only to the initial database and unique names axioms for actions.* The foundational axioms for the situation calculus, and the action precondition and successor state axioms are not required (although, of course, these last two are used in the regression steps).

4 An Implementation

We have implemented (in Eclipse Prolog) a narrative compiler and query evaluator based on Theorem 3.1 for procedure-free GOLOG programs, and we briefly describe it here.[6]

Given basic action theory $\mathcal{D}$, query Q and narrative ν, our task is to establish the entailment $\mathcal{D} \models (\forall s).Do(\nu, S_0, s) \supset Q(s)$. By Theorem 3.1, this is equivalent to establishing, for $i = 1, \ldots, n$, that $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models (\forall \vec{x}_i).\mathcal{R}[C_i] \supset \mathcal{R}[Q(\sigma_i)]$. Skolemize the leading universal quantifiers to obtain the following equivalent theorem proving task:

[6] The full program, together with supporting code for narratives about the blocks world based on the axioms of Example 3.1, is available on request from the author.

$\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}[C_i^{sk}] \supset \mathcal{R}[Q(\sigma_i^{sk})]$ for $i = 1, \ldots, n$. Here, C_i^{sk} and σ_i^{sk} are the results of substituting distinct, fresh Skolem constants for the free occurrences of $\vec{x}_i$ in C_i and σ_i, respectively.[7] Finally, this theorem proving task is equivalent to establishing that

$$\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \cup \{\mathcal{R}[C_i^{sk}]\} \models \mathcal{R}[Q(\sigma_i^{sk})] \text{ for } i = 1, \ldots, n. \quad (4)$$

The implementation has three components: a *narrative compiler*, a *query processor*, and a *theorem prover*.

4.1 The Narrative Compiler

This accepts a narrative program ν as described in Lemma 3.2, and performs the following steps:

1. Convert ν to its choice prenex form κ by applying the transformations specified in the proof of Lemma 3.1. Then determine $DoCpf(\kappa, S_0, s)$ to obtain the sentence (3).
2. Replace the variables $\vec{x}_i$ mentioned by C_i and σ_i in (3) by Skolem constants, yielding C_i^{sk} and σ_i^{sk} as in (4).
3. Regress C_i^{sk}, then convert this to clausal form.
4. Transform the sentences of $\mathcal{D}_{S_0}$ to clausal form. (The unique names axioms are not converted to clausal form; instead, these are represented by suitable simplification rules used by the regression routine and the theorem-prover described below.) Typically, $\mathcal{D}_{S_0}$ will include all of the domain's state constraints, relativized to S_0. For the blocks world, these consist of:

$$(\forall x, y).on(x, y, S_0) \supset \neg on(y, x, S_0),$$
$$(\forall x, y, z).on(x, y, S_0) \wedge on(x, z, S_0) \supset y = z,$$
$$(\forall x, y, z).on(x, z, S_0) \wedge on(y, z, S_0) \supset x = y.$$

The end result of these four steps is n distinct databases of clauses.

4.2 The Query Processor

For $i = 1, \ldots, n$ the query processor takes the query $Q(s)$, substitutes σ_i^{sk} for s as in (4), regresses $\neg Q(\sigma_i^{sk})$, converts this to clausal form, and adds the resulting clauses to the i-th database created by the narrative compiler. To establish the entailment (4), each of the n resulting sets of clauses must be shown to be unsatisfiable, and this computation is performed by the theorem-prover.

4.3 The Theorem-Prover

This is a relatively unsophisticated, incomplete unit resolution-based system, using subsumption for clause elimination. It also incorporates a limited form of equality reasoning.[8] The theorem-prover works in two passes:

1. It first tries a pure unit resolution refutation. In doing so, it takes equality into account as follows: Whenever it derives a ground unit clause of the form $t_1 = t_2$, it uniformly substitutes t_2 for t_1 throughout the current set of clauses, and also performs routine simplifications like replacing atoms of the form $t = t$ by *true* and $\neg t = t$ by *false*. Should it find a refutation this way, it exits with success.
2. Otherwise, the theorem-prover tries a little bit harder by performing a case analysis. It does this by repeatedly selecting and splitting a non-unit clause with at most one non-ground literal, and attempting a unit resolution refutation as in 1 for each of the cases. If this case analysis succeeds for some splittable clause, it exits with success; else it gives up.

4.4 An Example Program Execution

The following is the output obtained for compiling the ongoing narrative of Example 2.1, and issuing the query

$$Q(s) = (\exists x).onTable(x, s) \wedge (\exists y).on(A, y, s) \wedge x \neq y.$$

Example: Compiling and Querying a Narrative.[9]

```
[eclipse 2]: compile((move(a,b) # move(a,c)) :
                pi(x,?(onTable(x)) : move(d,x))).

Time (sec): 0.06

yes.
[eclipse 3]: prove(some(x,onTable(x) &
                    some(y,on(a,y) & -(x = y)))).
```

[7] Since s is the only free variable mentioned by a query $Q(s)$, and since the bound variables of $Q(s)$ are assumed distinct from the $\vec{x}_i$ (see statement of Theorem 3.1), the only way that $Q(\sigma_i)$ can mention one or more of the variables $\vec{x}_i$ is if σ_i mentions these variables.

[8] We did not incorporate time into the implementation, as would be needed for examples like 6 of Section 2. This would require special purpose temporal reasoning mechanisms in the theorem-prover, and our primary objective here was only to demonstrate the basic feasibility of our approach.

[9] CPU times here are for a SUN Enterprise (Ultra) 450, with four 400MHZ processors and 4GB of RAM.

```
Case: do(move(d, sk(13)), do(move(a, c), s0))

Unit resolution fails. Trying harder...

Splitting on [sk(13) = d, sk(13) = c]

  Succeeds.

Case: do(move(d, sk(12)), do(move(a, b), s0))

Unit resolution fails. Trying harder...

Splitting on [sk(12) = d, sk(12) = b]

  Succeeds.

*** Proof succeeds ***

Time (sec): 0.17

yes.
```

5 Discussion

We have so far avoided discussing concurrency, as would be required, for example, by even a simple narrative like: Pat moved blocks A and B to the table. One way to represent this as a program is with interleaving:

$$moveToTable(A)\,;\, moveToTable(B) \,|\, moveToTable(B)\,;\, moveToTable(A).$$

But this treatment of the *moveToTable* action is too course grained; it precludes the possibility of the two actions overlapping. A finer grained representation can be obtained by introducing *process fluents* (moving a block to the table) and two instantaneous actions, one to initiate the process, one to terminate it (See the discussion in [16]). With this representational device in hand, we can write a GOLOG program that describes all the possible interleavings of the two move actions (no overlap with A preceding/following B; partial overlap with A starting first, then B starting, then A ending, then B ending; total inclusion, with B starting, then A starting, then A ending, then B ending; etc). The combinatorics are bad enough, even for this simple example; they quickly get out of hand for more interesting examples like McCarthy's Junior-goes-to-Moscow [9]. They become impossible when the temporal ordering between *programs* is underspecified by a narrative; then we need an account for the concurrent execution of arbitrary GOLOG programs. This is exactly what the programming language CONGOLOG provides [4], so to accommodate concurrency, we can generalize our narratives-as-programs viewpoint by representing narratives as CONGOLOG programs. To define the result of querying a narrative, we appeal to CONGOLOG's $Trans^*$ predicate in Definition 2.2 instead of GOLOG's *Do* relation. Querying a narrative now becomes formally identical to proving properties of concurrent programs. These are all issues that have yet to be explored.

Our theoretical and implementation results are for procedure-free programs. It would not be difficult to incorporate non-recursive procedures into our account, by simply "unfolding" procedure calls. But of course, this would make sense only when this unfolding is guaranteed to terminate, namely, when there are no recursive calls. Since the definition of while loops of Section 2.1 requires recursion, our approach cannot deal with loops either. Providing computational foundations for loops and recursive procedures requires reasoning about program *postconditions*, and it appears that regression is not a suitable mechanism for this.

The cardinal sin of omission of this paper is to have glossed over how one translates a natural language narrative into a program. This raises a variety of issues in nonmonotonic reasoning concerned with minimizing action occurrences (e.g. [18]), but exactly how these techniques might be adapted to the automatic generation of programs is a completely open problem. Incidentally, this is not a consequence of our commitment to the situation calculus; any action logic adopting our view that narratives are programs must confront this problem.

References

[1] J.F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[2] C. Baral, A. Gabaldon, and A. Provetti. Formalizing narratives using nested circumscription. *Artificial Intelligence*, 104:107–164, 1998.

[3] M. Gelfond and V. Lifschitz. Action languages. *Linköping Electronic Articles in Computer and Information Sciences*, 3, 1998. www.ep.liu.se/ea/cis/1998/016/.

[4] G. De Giacomo, Y. Lespérance, and H.J. Levesque. Reasoning about concurrent execution, prioritized interrupts, and exogenous actions in the situation calculus. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 1221–1226, Nagoya, Japan, 1997.

[5] L. Karlsson. Anything can happen: On narratives and hypothetical reasoning. In A.G. Cohn and L.K. Schubert, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR'98)*, pages 36–47. Morgan Kaufmann Publishers, San Francisco, CA, 1998.

[6] R.A. Kowalski and M.J. Sergot. A logic-based calculus of events. *New Generation Computing*, 4:267, 1986.

[7] H.J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl. GOLOG: a logic programming language for dynamic domains. *J. of Logic Programming, Special Issue on Actions*, 31(1-3):59–83, 1997.

[8] J. McCarthy. Situations, actions and causal laws. Technical report, Stanford University, 1963. Reprinted in Semantic Information Processing (M. Minsky ed.), MIT Press, Cambridge, Mass., 1968, pp. 410-417.

[9] J. McCarthy and T. Costello. Combining narratives. In A.G. Cohn and L.K. Schubert, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR'98)*, pages 48–59. Morgan Kaufmann Publishers, San Francisco, CA, 1998.

[10] R. Miller and M. Shanahan. Narratives in the situation calculus. *The Journal of Logic and Computation (Special Issue on Actions and Processes)*, 4:513–530, 1994.

[11] J.A. Pinto and R. Reiter. Reasoning about time in the situation calculus. *Annals of Mathematics and Artificial Intelligence*, 14(2-4):251–268, September 1995.

[12] Javier Pinto. Occurrences and narratives as constraints in the branching structure of the situation calculus. *Journal of Logic and Computation*, 8(6):777–808, 1998.

[13] F. Pirri and R. Reiter. Some contributions to the metatheory of the situation calculus. *Journal of the ACM*, 46(3):261–325, 1999.

[14] R. Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, San Diego, CA, 1991.

[15] R. Reiter. Natural actions, concurrency and continuous time in the situation calculus. In L.C. Aiello, J. Doyle, and S.C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR'96)*, pages 2–13. Morgan Kaufmann Publishers, San Francisco, CA, 1996.

[16] R. Reiter. Sequential, temporal GOLOG. In A.G. Cohn and L.K. Schubert, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR'98)*, pages 547–556. Morgan Kaufmann Publishers, San Francisco, CA, 1998.

[17] E. Sandewall. *Features and Fluents: The Representation of Knowledge about Dynamical Systems.* Oxford University Press, 1994.

[18] M.P. Shanahan. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia.* MIT Press, 1997.

Representing the Knowledge of a Robot

Michael Thielscher
Department of Computer Science
Dresden University of Technology
01062 Dresden (Germany)
mit@inf.tu-dresden.de

Abstract

Acquiring information about its environment by sensing is a crucial ability of autonomous robots. Based on the established solution to the Frame Problem of the Fluent Calculus, we present a new, unifying formalism for representing and reasoning about sensing actions, knowledge preconditions, conditional actions, non-knowledge, and about what goals a robot can possibly achieve.

1 INTRODUCTION

Intelligent, autonomous robots choose most of their actions conditioned on the state of their environment. They are equipped with sensors for the purpose of acquiring information about the external world. Robots that perform reasoning about their goals and actions thus need an explicit representation of what they know of a state and how sensing affects their knowledge [Moore, 1985]. For example, the state of a door, that is, whether open or closed, should be known to a robot before it tries to enter the room behind. If the robot does not know then it should plan ahead both the appropriate sensing action and the possibility of having to open the door. This requires the robot to reason about what it currently knows or does not know, and what it will know after sensing.

A formal account of robot knowledge is also needed to prove the ability of a robot to achieve certain goals [Lin and Levesque, 1997]. For example, suppose that the door in question is closed but that its state can be altered by pressing a button next to it. Nonetheless, a robot, call it *Blindie*, who is unable to sense the state of the door will not be able, without assistance, to achieve the goal of entering the room. For it can never know whether it should press the button or not. Likewise unable to achieve the goal will be a robot, call it *Dumbie*, who can see but does not know how the button is causally related to the door. For this robot cannot conclude that it simply must press the button. The latter conclusion shows that a general account of the knowledge of robots must allow for distinguishing between the actual effects of actions and what a robot knows about these effects—an aspect which is not covered by existing accounts of knowledge in action formalisms, such as [Scherl and Levesque, 1993; Lobo *et al.*, 1997; Son and Baral, 1998].

The contribution of the present paper to the research on knowledge in cognitive robotics is manifold:

1. We extend the Fluent Calculus solution to the Frame Problem of [Thielscher, 1999], that is, the concept of state update axioms, to representing and reasoning about the knowledge of a robot.

2. Our theory provides a solution not only to the representational but also the inferential ([Bibel, 1998]) Frame Problem for knowledge.

3. Our theory enables the definition of conditional actions which can be inserted into a plan by a deductive planner.

4. By following a simple axiomatization scheme, our theory can readily be used to reason about what a robot does *not* know; unlike the approach of [Lakemeyer and Levesque, 1998], there is no need for a non-standard semantics or complex second-order axioms to this end.

5. Distinguishing between the actual effects of actions and what a robot knows about them, our theory supports an account of goal achievability *within* the corresponding Fluent Calculus axiomatization, as opposed to the meta-notion of achievability of [Lin and Levesque, 1997].

The key to these achievements lies in the explicit notion of a state offered by the Fluent Calculus aside from the standard notion of a situation. Thus the distinguishing feature of our theory is to formalize the knowledge of a robot in terms of possible *states* rather than possible *situations*.

The organization of the paper is as follows. In the next section, we give an informal overview of our approach to formalizing the knowledge of a robot. In Section 3, we briefly repeat the formal notions and notations of the basic Fluent Calculus. In Section 4, we extend this calculus to the representation of knowledge of states. We formally introduce the concept of knowledge update axioms and prove some crucial properties of our approach. In Section 5, we show how our approach provides a simple and elegant way of reasoning about what a robot does *not* know. In Section 6, we show how our approach can be used to reason about the ability of robots. We summarize in Section 7.

2 KNOWLEDGE IN THE FLUENT CALCULUS: THE BASIC IDEA

The knowledge a robot has of its environment shall be represented using a new binary predicate denoted by $KState(s, z)$. This relation is meant to hold iff in situation s the robot considers z a possible world state.[1] To specify what a robot knows of a particular situation S, an axiom of the form $(\forall z)\,(KState(S, z) \supset \Phi(z))$ is used, where sub-formula Φ constrains z according to the knowledge the robot has about z. Let $Holds(f, z)$ denote that fluent f holds in state z, then an example is,[2]

$$KState(S_0, z) \supset (\exists x)\,(Door(x, Room411) \wedge \neg Holds(Closed(x), z))$$

where $Door(x, y)$ is a static (non-fluent) predicate saying that door x leads to room y, and $Closed(x)$ denotes the fluent of door x being closed. Thus the implication says that in all states which are considered possible in situation S_0, there is a door to *Room411* which is not closed. On this basis, we can say that a robot *knows* that a fluent holds or does not hold in a situation iff the fluent either holds or does not hold in all possible states:[3]

$$Knows(f, s) \stackrel{\text{def}}{=} (\forall z)\,(KState(s, z) \supset Holds(f, z))$$
$$Knows(\neg f, s) \stackrel{\text{def}}{=} (\forall z)\,(KState(s, z) \supset \neg Holds(f, z))$$

These abbreviations generalize to the knowledge of compound formulas in a natural way.

The macro *Knows* can be used to specify knowledge preconditions for actions. For instance, let $Poss(a, s)$ denote that a is possible in situation s, then according to the following axiom a robot must know of some open door if it wants to enter a room:

$$Poss(Enter(y), s) \equiv (\exists x)\,(Door(x, y) \wedge Knows(\neg Closed(x), s))$$

The representational Frame Problem for knowledge is solved by axioms that specify the relation between the possible states before and after an action: The effect of an action $A(\vec{x})$, be it sensing or not, on the knowledge of a robot is specified by a *knowledge update axiom*,

$$\Delta \supset (\forall z)\,(KState(Do(A(\vec{x}), s), z) \equiv (\exists z')(KState(s, z') \wedge \Psi(\vec{x}, z, z', s))) \quad (1)$$

where $Do(a, s)$ denotes the situation reached by performing a in s and sub-formula Ψ defines how, under condition Δ, the states considered possible after the action, z, relate to the states considered possible beforehand, z'. Let, for example, $SenseDoor(x)$ denote the action of sensing whether a door x is closed, then this is a suitable knowledge update axiom:[4]

$$Poss(SenseDoor(x), s) \supset (\forall z)\,(KState(Do(SenseDoor(x), s), z) \equiv KState(s, z) \wedge [Holds(Closed(x), z) \equiv Holds(Closed(x), s)]) \quad (2)$$

where $Holds(f, s)$ means that fluent f actually holds in situation s. Hence, the axiom says that among the states possible in s only those are still considered possible after sensing which agree with the actual state of the world as far as the status of the sensed door is concerned. A crucial immediate consequence of (2) in the light of the above definition of *Knows* is that a

[1] For the reader who is unfamiliar with the Fluent Calculus we note that states are reified, i.e., first-class citizens; see Section 3 for the details.

[2] A word on the notation: Predicate and function symbols, including constants, start with a capital letter whereas variables are in lower case, sometimes with sub- or superscripts. Free variables in formulas are assumed universally quantified. Throughout the paper, action variables are denoted by the letter a, situation variables by the letter s, fluent variables by the letter f, and state variables by the letter z, all possibly with sub- or superscript.

[3] Throughout the paper we assume that the knowledge of a robot—though incomplete—is correct, that is, the actual world state is always among the states which are considered possible. (Foundational axiom (11) in Section 4.2 below expresses this assumption formally.)

[4] The knowledge update axiom below is of the form $\Delta \supset (\forall z)\,(KState(Do(a, s), z) \equiv KState(s, z) \wedge \Psi)$, whose equivalent correct form is $\Delta \supset (\forall z)\,(KState(Do(a, s), z) \equiv (\exists z')\,(KState(s, z') \wedge \Psi \wedge z = z'))$.

robot capable of sensing knows afterwards whether the door is open:

$$Poss(SenseDoor(x), s) \supset Knows(Closed(x), Do(SenseDoor(x), s)) \vee Knows(\neg Closed(x), Do(SenseDoor(x), s))$$

(This follows since only one of $Holds(Closed(x), s)$ and $\neg Holds(Closed(x), s)$ can be true.)

Providing a solution to the representational aspect of the Frame Problem for knowledge, knowledge update axioms lay the foundations for overcoming the inferential aspect, too. The inference scheme employed to this end takes as input an implication of the form $KState(\sigma, z) \supset \Phi(z)$, which specifies the knowledge of a situation σ, along with the consequence of an aplicable knowledge update axiom of the form (1) for an action $A(\vec{\tau})$. Then an immediate logical consequence of the two input formulas is,

$$KState(Do(A(\vec{\tau}), \sigma), z) \supset (\exists z')(\Phi(z') \wedge \Psi(\vec{x}, z, z', s))$$

which provides a full specification of what is known about the successor situation $Do(A(\vec{\tau}), \sigma)$. There is no need to carry over to the new situation all pieces of knowledge one-by-one and using separate instances of axioms, which shows why and how the inferential Frame Problem can be solved on the basis of knowledge update axioms.

In case of non-sensing actions, knowledge update axioms describe what a robot knows of their effect. Examples are given below, in Sections 4 and 6. Since these specifications are independent of state update axioms, which describe the actual effect of actions, it is possible to formally represent and reason about limitations of a robot as regards its knowledge of the dynamics of the environment.

3 THE SIMPLE FLUENT CALCULUS: STATE UPDATE AXIOMS

In the following we provide a brief description of the fundamentals of the Fluent Calculus; for a complete introduction see [Thielscher, 1999] or the electronically available, archived reference article [Thielscher, 1998].[5] The Fluent Calculus, which roots in the logic programming formalism of [Hölldobler and Schneeberger, 1990], is an order-sorted second order language with equality, which includes the sorts *action*, *sit*, *fluent*, and *state*, with *fluent* being a sub-sort of *state*. Fluents are reified propositions. That is to say, terms like, for instance, $Closed(x)$ denote fluents, where *Closed* is a unary function symbol. Fluents can be joined together by the binary function symbol "$\circ$" to make up states. We write this symbol in infix notation. The function shall satisfy the laws of associativity and commutativity and admit a unit element, denoted by $\emptyset$. Associativity allows us to omit parentheses in nested applications of $\circ$.

A function $State : sit \mapsto state$ relates a situation to the state of the world in that situation. As an example, consider three doors *Door1*, *Door2*, and *Door3*, the first of which is initially closed while the second one is open. Moreover, the robot is currently not in front of *Door1* or *Door3* but in front of *Door2*. Let the fluents $Closed(x)$ and $InFrontOf(x)$ denote, resp., that door x is closed and that the robot is in front of door x. Then the given incomplete specification (nothing is said about *Door3* being open or not) of the initial situation, S_0, can be axiomatized in the Fluent Calculus as follows:

$$\begin{aligned}(\exists z)[\, State(S_0) = {} & Closed(Door1) \\ & \circ InFrontOf(Door2) \circ z \\ \wedge\ (\forall z')(\, & z \neq Closed(Door2) \circ z' \wedge \\ & z \neq InFrontOf(Door1) \circ z' \wedge \\ & z \neq InFrontOf(Door3) \circ z')\,]\end{aligned} \tag{3}$$

That is, of the initial state $State(S_0)$ it is known that both $Closed(Door1)$ and $InFrontOf(Door2)$ are true and that possibly some other fluents z hold except for each of $Closed(Door2)$, $InFrontOf(Door1)$, and $InFrontOf(Door3)$, of which we know they are not true in S_0.

Fundamental for any Fluent Calculus axiomatization is the axiom set *EUNA*, which extends given unique name-assumptions by the axioms AC1 (i.e., associativity, commutativity, and unit element):

$$\begin{aligned}(z_1 \circ z_2) \circ z_3 &= z_1 \circ (z_2 \circ z_3) \\ z_1 \circ z_2 &= z_2 \circ z_1 \\ z \circ \emptyset &= z\end{aligned}$$

along with the following axioms, which entail inequality of state terms (as used, e.g., in (3)) if some fluent occurs in one but not in the other state:[6]

$$z = f \supset z \neq \emptyset \wedge [z = z' \circ z'' \supset z' = \emptyset \vee z'' = \emptyset]$$

$$z_1 \circ z_2 = z_3 \circ z_4 \supset (\exists z_a, z_b, z_c, z_d)[z_1 = z_a \circ z_b \wedge z_2 = z_c \circ z_d \wedge z_3 = z_a \circ z_c \wedge z_4 = z_b \circ z_d]$$

[5] or see the online tutorial at `http://pikas.inf.tu-dresden.de/~mit/FC/Tutorial/index.htm`

[6] Unlike the definition of *EUNA* in terms of unification completeness wrt. AC1, as used in earlier versions of the Fluent Calculus [Hölldobler and Thielscher, 1995; Thielscher, 1999], the new axioms allow to incorporate domain dependent assumptions of unique names [Störr and Thielscher, 2000].

For computing with state terms, two immediate consequences of these axioms are of importance, the following rules of cancellation and distribution.

Proposition 1 [Störr and Thielscher, 2000] *Axioms EUNA entail:*

1. *If* $z_1 \circ f = z_2 \circ f$, *then* $z_1 = z_2$.
2. *If* $f_1 \neq f_2$ *and* $z_1 \circ f_1 = z_2 \circ f_2$, *then* $(\exists z')\, z_2 = f_1 \circ z'$ *and* $(\exists z')\, z_1 = f_2 \circ z'$.

In addition, we have the foundational axiom

$$State(s) \neq f \circ f \circ z \tag{4}$$

by which double occurrences of fluents are prohibited in any state which is associated with a situation. (It will be explained shortly why "$\circ$" is not required to be idempotent to this end.) Finally, the Fluent Calculus uses the expressions $Holds(f, z)$ —denoting that f holds in state z—and the common $Holds(f, s)$—stating that fluent f holds in situation s—, though not as part of the signature but as mere abbreviations of equality sentences:

$$\begin{array}{l} Holds(f, z) \stackrel{\text{def}}{=} (\exists z')\, z = f \circ z' \\ Holds(f, s) \stackrel{\text{def}}{=} Holds(f, State(s)) \end{array} \tag{5}$$

So-called state update axioms specify the entire relation between the states at two consecutive situations. Deterministic actions with only direct and closed effects[7] give rise to the simplest form of state update axioms, where a mere equation relates a successor state $State(Do(A, s))$ to the preceding state $State(s)$:

$$\begin{array}{l} Poss(A(\vec{x}), s) \wedge \Delta(\vec{x}, s) \supset \\ \quad (\exists \vec{y})\, State(Do(A(\vec{x}), s)) \circ \vartheta^- = State(s) \circ \vartheta^+ \end{array} \tag{6}$$

where ϑ^- are the negative effects and ϑ^+ the positive effects, resp., of action $A(\vec{x})$ under condition $\Delta(\vec{x}, s)$ (and where $\vec{y}$ are the variables in ϑ^-, ϑ^+ which are not among $\vec{x}$).[8] While actions may have conditional effects, and hence multiple state update axioms, there is an assumptions generally made, namely, that a set of state update axioms be consistent and complete, in the following sense. Let Ax be a domain axiomatization, then for any action $A(\vec{x})$ the following holds: First, for any two different axioms (6) in Ax for action $A(\vec{x})$ and with conditions $\Delta_1(\vec{x}, s)$ and $\Delta_2(\vec{x}, s)$, we have

$$EUNA \models \neg[Poss(A(\vec{x}), s) \wedge \Delta_1(\vec{x}, s) \wedge \Delta_2(\vec{x}, s)] \tag{7}$$

(That is, no two different state update axioms for one action apply in the same situation.) Second, if $\Delta_1(\vec{x}, s), \ldots, \Delta_n(\vec{x}, s)$ are the conditions of all state update axioms (6) in Ax for action $A(\vec{x})$, then

$$Ax \models Poss(A(\vec{x}), s) \supset \Delta_1(\vec{x}, s) \vee \ldots \vee \Delta_n(\vec{x}, s) \tag{8}$$

(That is, there is always an applicable state update axiom for an action that is possible.)

Moreover, we assume that each action $A(\vec{x})$ is accompanied by a precondition axiom of the form,

$$Poss(A(\vec{x}), s) \equiv \pi(\vec{x}, s) \tag{9}$$

where π is a first-order formula without predicate $Poss$ and with free variables among $\vec{x}, s$ and in which s is the only term of sort sit.

As an example, let $Press(Button(x))$ denote the action of pressing the button next to door x, by which the door opens if it is closed and closes if it is open. This is a suitable pair of state update axioms for this action:

$$\begin{array}{l} Poss(Press(Button(x)), s) \wedge Holds(Closed(x), s) \\ \quad \supset State(Do(Press(Button(x)), s)) \circ Closed(x) \\ \qquad = State(s) \\ Poss(Press(Button(x)), s) \wedge \neg Holds(Closed(x), s) \\ \quad \supset State(Do(Press(Button(x)), s)) \\ \qquad = State(s) \circ Closed(x) \end{array} \tag{10}$$

That is to say, if x is currently closed then $Closed(x)$ becomes false whereas it becomes true if x is currently open. Let the precondition of $Press(Button(x))$ be given by the axiom $Poss(Press(Button(x)), s) \equiv Holds(InFrontOf(x), s)$. Suppose a scenario where the initial situation is described by formula (3), and consider the action of pressing the button next to $Door2$. The result can be inferred using the instance $\{x/Door2, s/S_0\}$ of the second one of our state update axioms (10): After verifying that $Poss(Press(Button(Door2)), S_0)$ and $\neg Holds(Closed(Door2), S_0)$, we can replace the expression $State(S_0)$ in the entailed equation by the term which equals $State(S_0)$ according to (3). So doing yields—setting $S_1 = Do(Press(Button(Door2)), S_0)$ and repeating the relevant additional information in (3) about z—,

$$\begin{array}{l} (\exists z)\, [\, State(S_1) = Closed(Door1) \circ InFrontOf(Door2) \\ \qquad \circ z \circ Closed(Door2) \\ \quad \wedge (\forall y, z')\, (z \neq InFrontOf(Door1) \circ z' \wedge \\ \qquad z \neq InFrontOf(Door3) \circ z')\,] \end{array}$$

[7] By closed effects we mean that an action does not have an unbounded number of direct effects.

[8] This scheme is the reason for not stipulating that "$\circ$" be idempotent, contrary to what one might intuitively expect. For if the function were idempotent, then the equation would not imply that $State(Do(A, s))$ does not include ϑ^-.

We thus obtain from an incomplete initial specification a still partial description of the successor state, which in particular includes the unaffected fluents $Closed(Door1)$ and $InFrontOf(Door2)$ and the still valid information about the robot not being in front of the two doors $Door1$ and $Door3$. Hence, all these pieces of information survived the computation of the effect of the action and so need not be carried over by separate application of axioms. This illustrates why and how state update axioms provide a solution not only to the representational but also the inferential Frame Problem.

Under the provision that actions have only closed effects, state update axioms of the form (6) can be fully mechanically generated from a set of simple Situation Calculus-style effect axioms if the latter can be assumed to provide a complete account of the relevant effects of an action. It has been proved that a collection of thus generated state update axioms correctly reflects the fundamental assumption of persistence. This is the primary theorem of the simple Fluent Calculus [Thielscher, 1999].

4 AXIOMATIZING KNOWLEDGE UPDATE

4.1 Extending the signature

The only addition to the signature of the basic Fluent Calculus required to represent knowledge, is the predicate

$$KState : sit \times state$$

with the intended meaning that according to the robot's knowledge the second argument is a possible state in the situation denoted by the first argument. On this basis, the fact that some property of a situation is known to the robot is specified using the macro $Knows$, which is defined as follows:

$$Knows(\varphi, s) \stackrel{\text{def}}{=} (\forall z)\,(KState(s, z) \supset HOLDS(\varphi, z))$$

where

$$\begin{aligned} HOLDS(f, z) &\stackrel{\text{def}}{=} Holds(f, z) \\ HOLDS(\neg\varphi, z) &\stackrel{\text{def}}{=} \neg HOLDS(\varphi, z) \\ HOLDS(\varphi \wedge \psi, z) &\stackrel{\text{def}}{=} HOLDS(\varphi, z) \wedge HOLDS(\psi, z) \\ HOLDS((\forall x)\,\varphi, z) &\stackrel{\text{def}}{=} (\forall x)\,HOLDS(\varphi, z) \end{aligned}$$

4.2 Foundational axioms

The Fluent Calculus with knowledge requires the addition of two foundational axioms, which characterize properties of the knowledge predicate. First, the knowledge of a robot is correct:

$$KState(s, State(s)) \tag{11}$$

(That is, the actual state of the world is always among the states considered possible.) Second, no possible state contains multiple occurrences of fluents (c.f. foundational axiom (4) of the simple Fluent Calculus):

$$KState(s, z) \supset (\forall f, z')\, z \neq f \circ f \circ z' \tag{12}$$

4.3 Knowledge update axioms

While state update axioms specify the effect of actions on the external world, update axioms for knowledge specify their effect on what the robot knows about the world.

Definition 2 A *knowledge update axiom* for an action $A(\vec{x})$ takes the form

$$\begin{aligned} &\Delta(\vec{x}, s) \supset \\ &\quad (\forall z)\,(KState(Do(A(\vec{x}), s), z) \equiv \\ &\qquad (\exists z')(KState(s, z') \wedge \Psi(\vec{x}, z, z', s)))\end{aligned}$$

where Δ and Ψ are first-order formulas with free variables among $\vec{x}, s$ and $\vec{x}, z, z', s$, resp.

Example 1 Consider a Fluent Calculus signature with fluents $InFrontOf(x)$, $Closed(x)$ and actions $SenseDoor(x)$, $Press(Button(x))$ with the obvious meaning. The robot may sense a door or press the button next to it iff it knows that it is in front of the door:

$$\begin{aligned} &Poss(SenseDoor(x), s) \equiv \\ &\qquad Knows(InFrontOf(x), s) \\ &Poss(Press(Button(x)), s) \equiv \\ &\qquad Knows(InFrontOf(x), s) \end{aligned} \tag{13}$$

Being a sensing action, $SenseDoor$ has no effect on the external world, hence the simple state update axiom

$$\begin{aligned} &Poss(SenseDoor(x), s) \supset \\ &\quad State(Do(SenseDoor(x), s)) = State(s) \end{aligned}$$

The action does, however, affect the knowledge of the robot in that the actual status of the door becomes known, as specified by knowledge update axiom (2) of Section 2.

Pressing the button next to a door causes a closed door to open and an open door to close; hence the state update axiom (10) of Section 3. If the robot knows about this effect, then the appropriate knowledge update axiom mirrors the actual update:

$$\begin{aligned} &Poss(Press(Button(x)), s) \supset \\ &\quad (\forall z)\,(KState(Do(Press(Button(x)), s), z) \equiv \\ &\qquad (\exists z')\,(KState(s, z') \wedge \\ &\quad [Holds(Closed(x), z') \supset z \circ Closed(x) = z'] \wedge \\ &\quad [\neg Holds(Closed(x), z') \supset z = z' \circ Closed(x)])) \end{aligned} \tag{14}$$

Put in words, any previously possible state z' in which door x is closed is considered possible after pressing the button if z' is modified by making false $Closed(x)$, and any previously possible state z' in which door x is open is considered possible after pressing the button if z' is modified by adding $Closed(x)$. ■

Our example illustrates two kinds of actions with special properties as regards knowledge, namely, pure sensing actions and actions of whose effects the agent has accurate knowledge.

Definition 3 Consider a set of state and knowledge update axioms Ax.

1. An action $A(\vec{x})$ is *pure sensing* in Ax if it has a single state update axiom and if this axiom is of the form

$$Poss(A(\vec{x}), s) \supset State(Do(A(\vec{x}), s)) = State(s)$$

2. Let

$$\begin{aligned} &Poss(A(\vec{x}), s) \wedge \Delta_1(\vec{x}, s) \supset \\ &\quad (\exists \vec{y}_1)\, State(Do(A(\vec{x}), s)) \circ \vartheta_1^- = State(s) \circ \vartheta_1^+ \\ &\qquad\qquad \vdots \\ &Poss(A(\vec{x}), s) \wedge \Delta_n(\vec{x}, s) \supset \\ &\quad (\exists \vec{y}_n)\, State(Do(A(\vec{x}), s)) \circ \vartheta_n^- = State(s) \circ \vartheta_n^+ \end{aligned}$$

be all state update axioms for an action $A(\vec{x})$, then $A(\vec{x})$ is *accurately known* in Ax if the following is the unique knowledge update axiom for this action:[9]

$$\begin{aligned} &Poss(A(\vec{x}), s) \supset \\ &\quad (\forall z)\,(KState(Do(A(\vec{x}), s), z) \equiv \\ &\qquad (\exists z')(KState(s, z') \wedge Poss(A(\vec{x}), z') \wedge \\ &\qquad\quad [\,\Delta_1(\vec{x}, z') \supset (\exists \vec{y}_1)\, z \circ \vartheta_1^- = z' \circ \vartheta_1^+\,] \wedge \\ &\qquad\qquad \vdots \\ &\qquad\quad [\,\Delta_n(\vec{x}, z') \supset (\exists \vec{y}_n)\, z \circ \vartheta_n^- = z' \circ \vartheta_n^+\,])) \end{aligned}$$

Our approach to robot knowledge enjoys two important properties as regards these two kinds of actions.

Theorem 4

1. *Pure sensing does not affect the world state.*
2. *For any situation s, any accurately known action a which is known to be possible in s, and any fluent f not affected by a, f is known to hold after performing a in s iff it is known to hold in s.*

Proof: The first claim follows immediately by definition of pure sensing actions.

For the second claim, observe first that if a is accurately known and possible in s, then the knowledge update axiom implies,

$$\begin{aligned} &(\forall z)\,(KState(Do(A(\vec{x}), s), z) \equiv \\ &\quad (\exists z')(KState(s, z') \wedge Poss(A(\vec{x}), z') \wedge \\ &\qquad [\,\Delta_1(\vec{x}, z') \supset (\exists \vec{y}_1)\, z \circ \vartheta_1^- = z' \circ \vartheta_1^+\,] \wedge \\ &\qquad\qquad \vdots \\ &\qquad [\,\Delta_n(\vec{x}, z') \supset (\exists \vec{y}_n)\, z \circ \vartheta_n^- = z' \circ \vartheta_n^+\,])) \end{aligned}$$

Since a is known to be possible in s, the assumptions of consistency and completeness (c.f. (7),(8)) imply that for each state z' satisfying $KState(s, z')$, there is a unique $i = 1, \ldots, n$ such that $\Delta_i(\vec{x}_i, z')$ is true. Let

$$(\exists \vec{y}_i)\, z \circ \vartheta_i^- = z' \circ \vartheta_i^+$$

be the equation implied by this $\Delta_i(\vec{x}_i, z')$. In turn, this equation implies $Holds(f, z)$ iff $Holds(f, z')$ according to the rule of distribution (Proposition 1). For f is not amongst the fluents in $\vartheta_i^-, \vartheta_i^+$ following the assumption that f is not affected by a. Hence, if f is true in all states possible in s, then f is true in all states possible in $Do(a, s)$. Conversely, if f is false in some state possible in s, then f is false in some state possible in $Do(a, s)$. ■

The first one of these two fundamental results coincides with a property of the Situation Calculus-based approach to sensing actions of [Scherl and Levesque, 1993], while the second one generalizes a property from [Scherl and Levesque, 1993] in that we additionally distinguish between the effect of the action itself and the robot's awareness of it. Note the necessity of the condition in Item 2 which requires the robot to know that the action is possible. For otherwise the mere executability of the action may provide the robot with new knowledge.

Item 2 shows that knowledge update axioms provide a solution to the representational Frame Problem for knowledge: Everything that is known before an action is performed is still known afterwards, provided it is known to being unaffected by the action. Knowledge update axioms moreover lay the foundations for overcoming the inferential aspect of the Frame Problem for knowledge, too. The following simple inference scheme can be employed to this end. Suppose

[9] Below, the expression $\Delta_i(\vec{x}, z)$ stands for $\Delta_i(\vec{x}, s)$ with each $Holds(f, s)$ replaced by $Holds(f, z)$; and the expression $Poss(A(\vec{x}), z)$ stands for $\pi(\vec{x}, s)$ with each $Holds(f, s)$ replaced by $Holds(f, z)$], where $\pi(\vec{x}, s)$ is the specification for $Poss(A(\vec{x}), s)$ (c.f. precondition schema (9)).

that the knowledge of the robot about a situation σ is given by $KState(\sigma, z) \supset \Phi(z)$. Suppose further an action $A(\vec{r})$, sensing or not, with knowledge update axiom (1) is performed in σ. Then an immediate logical consequence of the instance $\{\vec{x}/\vec{r}, s/\sigma\}$ of (1) and the implication just mentioned, is

$$KState(Do(A(\vec{r}), \sigma), z) \supset (\exists z')(\Phi(z') \wedge \Psi(\vec{r}, z, z', s))$$

(assuming successful evaluation of Δ), which provides a specification of what is known about the successor situation $Do(A(\vec{r}), \sigma)$. There is no need to carry over to the new situation all pieces of knowledge one-by-one and using separate instances of axioms.

Example 1 (continued) Suppose that of the initial situation the robot knows that it is in front of *Door1* and not in front of *Door2* and that *Door2* is closed, that is,

$$\begin{aligned} &KState(S_0, z) \supset \\ &\quad Holds(InFrontOf(Door1), z) \wedge \\ &\quad \neg Holds(InFrontOf(Door2), z) \wedge \\ &\quad Holds(Closed(Door2), z) \end{aligned} \tag{15}$$

Let $S_1 = Do(SenseDoor(Door1), S_0)$ and consider the instance $\{x/Door1, s/S_0\}$ of knowledge update axiom (2). Then the sub-formula $KState(S_0, z)$ of this instance can be replaced by its consequence as given in (15), which yields, after evaluating the antecedent against the precondition axiom in (13),

$$\begin{aligned} &KState(S_1, z) \supset \\ &\quad Holds(InFrontOf(Door1), z) \wedge \\ &\quad \neg Holds(InFrontOf(Door2), z) \wedge \\ &\quad Holds(Closed(Door2), z) \wedge \\ &\quad (Holds(Closed(Door1), z) \equiv \\ &\qquad Holds(Closed(Door1), S_0)) \end{aligned}$$

We thus obtain a description of what is known about the successor state, which in particular includes the unaffected knowledge about $InFrontOf(Door1)$, $InFrontOf(Door2)$, and $Closed(Door2)$. Hence, all this knowledge is still readily available. ■

4.4 Conditional actions

Employing a theory of sensing actions for robot planning is known to require more complex a notion of a plan than given by the classical view of plans as mere sequences of elementary actions [Levesque, 1996]. At the very least, a robot must be able to condition its course of actions on the result of a sensing action. This minimal requirement can be satisfied in our approach by introducing the concept of a *conditional* action, based on the function

$$If : fluent \times action \mapsto action$$

An instance $If(f, a)$ shall be interpreted as denoting action a if condition f is known to hold, otherwise as denoting the 'action' of doing nothing. A conditional action is defined as possible iff the truth value of the condition is known to the robot and if, provided the condition is true, the respective action is possible:

$$\begin{aligned} Poss(If(f, a), s) \equiv\ &[Knows(f, s) \vee Knows(\neg f, s)] \\ &\wedge [Knows(f, s) \supset Poss(a, s)] \end{aligned} \tag{16}$$

The effect of a conditional action on the world state and on the robot's knowledge state is identical to the effect of the action if it applies; otherwise, the conditional has no effect:

$$\begin{aligned} &Poss(If(f, a), s) \supset \\ &\quad [Knows(f, s) \supset \\ &\qquad State(Do(If(f, a), s)) = State(Do(a, s)) \wedge \\ &\qquad (\forall z)\,(KState(Do(If(f, a), s), z) \equiv \\ &\qquad\qquad KState(Do(a, s), z))] \wedge \\ &\quad [Knows(\neg f, s) \supset \\ &\qquad State(Do(If(f, a), s)) = State(s) \wedge \\ &\qquad (\forall z)\,(KState(Do(If(f, a), s), z) \equiv \\ &\qquad\qquad KState(s, z))] \end{aligned} \tag{17}$$

Example 2 Suppose that the robot knows it is in front of *Door1*. It is not given whether the door is open. Formally,

$$KState(S_0, z) \supset Holds(InFrontOf(Door1), z) \tag{18}$$

Let Ax denote the conjunction of this axiom, the precondition and the state and knowledge update axioms for $SenseDoor(x)$ and $Press(Button(x))$ from above, axioms (16) and (17) defining If, and the foundational axioms of the Fluent Calculus. The task shall be to find a plan after whose execution the robot knows that *Door1* is open. A solution is the term

$$\begin{aligned} \sigma = Do(&If(Closed(Door1), Press(Button(Door1))), \\ &Do(SenseDoor(Door1), S_0)) \end{aligned}$$

which satisfies $Ax \models Knows(\neg Closed(Door1), \sigma)$:

From (18) and (13), $Poss(SenseDoor(Door1), S_0)$. Let $S_1 = Do(SenseDoor(Door1), S_0)$, then from (2),

$$\begin{aligned} KState(S_1, z) \supset\ &KState(S_0, z) \wedge \\ &[Holds(Closed(Door1), z) \equiv \\ &\ Holds(Closed(Door1), S_0)] \end{aligned}$$

which implies both that $Knows(Closed(Door1), S_1) \vee Knows(\neg Closed(Door1), S_1)$ and, according to (18), $Knows(InFrontOf(Door1), S_1)$. By (16) and (13), $Poss(If(Closed(Door1), Press(Button(Door1))), S_1)$. Hence, from (17), (14), and (12) it follows that

$$\begin{aligned} &KState(Do(If(Closed(Door1), \\ &\qquad Press(Button(Door1))), S_1), z) \\ &\quad \supset \neg Holds(Closed(Door1), z) \end{aligned}$$

which implies $Ax \models Knows(\neg Closed(Door1), \sigma)$. ■

5 WHAT DOES A ROBOT NOT KNOW?

The explicit notion of a state in the Fluent Calculus for the representation of state knowledge offers an intriguingly simple and elegant way of reasoning about what a robot does *not* know, following the motivation of [Lakemeyer and Levesque, 1998]. To specify that a robot knows and only knows certain facts about the state of the world in a situation S, one employs an axiom of the form $KState(S, z) \equiv \Phi(z)$, where Φ describes all that is known about z. On this basis, it is straightforward to prove that, say, the truth value of some fluent f is *not* known in S by proving validity of $(\exists z)\,(\Phi(z) \wedge Holds(f, z)) \wedge (\exists z')\,(\Phi(z') \wedge \neg Holds(f, z'))$.

Example 3 Suppose the robot knows that in situation S_0 it is in front of two doors $Door1$ and $Door2$ and that it knows that at least one of them is not closed, but it does not know which one.[10] This combination of knowledge with ignorance is formally specified by,

$$\begin{array}{l} KState(S_0, z) \equiv \\ \quad Holds(InFrontOf(Door1), z) \wedge \\ \quad Holds(InFrontOf(Door2), z) \wedge \\ \quad [\neg Holds(Closed(Door1), z) \vee \\ \qquad \neg Holds(Closed(Door2), z)] \wedge \\ \quad (\forall f, z')\, z \neq f \circ f \circ z' \end{array} \tag{19}$$

Note that the last conjunct is necessary in order not to produce a logical contradiction to foundational axiom (12). Let Ax denote the conjunction of this axiom, the precondition and the state and knowledge update axioms for $SenseDoor(x)$ from above, and the foundational axioms of the Fluent Calculus. Then we can draw the following conclusions, which exemplify the interaction between knowing, not knowing, and sensing.

1. The robot knows that some door is open initially, that is,

   $$Ax \models Knows((\exists x)\,\neg Closed(x), S_0)$$

   This can be easily seen from the equivalent proposition

   $$Ax \models (\forall z)\,(KState(S_0, z) \supset (\exists x)\,\neg Holds(Closed(x), z))$$

   which follows directly from (19).

2. However, the robot does not know of any particular open door initially, that is,

   $$Ax \models \neg(\exists x)\,Knows(\neg Closed(x), S_0)$$

   This follows from the equivalent proposition

   $$Ax \models (\forall x)(\exists z)\,(KState(S_0, z) \wedge Holds(Closed(x), S_0))$$

   For in case $x = Door1$ the state

   $$z = InFrontOf(Door1) \circ InFrontOf(Door2) \circ Closed(Door1)$$

   satisfies the conjunct; in case $x = Door2$ the state

   $$z = InFrontOf(Door1) \circ InFrontOf(Door2) \circ Closed(Door2)$$

   satisfies the conjunct; and in case $x \neq Door1$ and $x \neq Door2$ the state

   $$z = InFrontOf(Door1) \circ InFrontOf(Door2) \circ Closed(Door1) \circ Closed(x)$$

   satisfies the conjunct.

3. After sensing the state of $Door1$, the robot will know of a particular open door (although it is not known in advance which one), that is,

   $$Ax \models (\exists x)\,Knows(\neg Closed(x), S_1)$$

   where $S_1 = Do(SenseDoor(Door1), S_0)$. This follows from the equivalent proposition

   $$Ax \models (\exists x)(\forall z)\,(KState(S_1, z) \supset \neg Holds(Closed(x), z))$$

   For, we have $Poss(SenseDoor(Door1), S_0)$ from (13) and (19). Hence, from (2) it follows that

   $$(\forall z)\,(KState(S_1, z) \equiv KState(S_0, z) \wedge \neg Holds(Closed(Door1), z))$$

   in case $\neg Holds(Closed(Door1), S_0)$, and

   $$(\forall z)\,(KState(S_1, z) \equiv KState(S_0, z) \wedge Holds(Closed(Door1), z))$$

   in case $Holds(Closed(Door1), S_0)$. Furthermore, from (19) we have

   $$(\forall z)\,(KState(S_0, z) \wedge Holds(Closed(Door1), z) \supset \neg Holds(Closed(Door2), z))$$

   Altogether, both if $\neg Holds(Closed(Door1), S_0)$ and if $Holds(Closed(Door1), S_0)$ there is some x such that for all z, $KState(S_1, z)$ implies $\neg Holds(Closed(x), z)$. ■

[10] This is an adaptation of the example of [Lakemeyer and Levesque, 1999].

With knowledge update axioms the inferential Frame Problem for 'only knowing' can be solved by employing an inference scheme analogous to the one presented in Section 4.3. Suppose that the given complete knowledge of the robot about a situation σ is specified by $KState(\sigma, z) \equiv \Phi(z)$, as described above. Suppose further an action $A(\vec{\tau})$, sensing or not, with update axiom (1) is performed in σ. Then an immediate logical consequence of the instance $\{\vec{x}/\vec{\tau}, s/\sigma\}$ of (1) and the equivalence just mentioned, is

$$KState(Do(A(\vec{\tau}), \sigma), z) \equiv (\exists z')(\Phi(z') \wedge \Psi(\vec{x}, z, z', s))$$

(assuming successful evaluation of Δ), which provides a complete specification of what is known and what is not known about the successor situation $Do(A(\vec{\tau}), \sigma)$. Again, there is no need to carry over to the new situation all pieces of knowledge and non-knowledge one-by-one and using separate instances of axioms.

6 REASONING ABOUT ABILITY

The independence of state update specifications from knowledge update specifications enables the ready usage of our formalism for the purpose of reasoning about possibly restricted subjective achievability. Two kinds of 'mental' limitations may prevent a robot from reaching a goal although it would be physically able to do so: The robot may lack crucial state knowledge without having at hand the appropriate sensing action, or it may lack complete knowledge of the effect of its actions.

Formal proofs of non-achievability rely on induction over situations along the line of [Reiter, 1993], where the following second-order axiom has been introduced:

$$(\forall \Pi)\,(\Pi(S_0) \wedge (\forall a, s)\,(\Pi(s) \supset \Pi(Do(a, s))) \supset (\forall s)\,\Pi(s))$$

That is, a property Π holds for all situations if it holds initially and if all actions preserve it. Usually, one is interested in proving properties only for those situations that are reachable by an executable sequence of actions. To this end, [Reiter, 1993] adds the following foundational axioms:

$$(\forall s) \neg s < S_0$$
$$(\forall a, s, s')\,(s < Do(a, s') \equiv Poss(a, s') \wedge s \leq s')$$

where $s \leq s'$ abbreviates $s < s' \vee s = s'$.

Using induction in the Fluent Calculus requires a domain closure axiom for actions since the induction step can never be proved if there are actions without state update axioms, in which case successor situation may enjoy arbitrary properties. Let $A_1(\vec{x}_1), \ldots, A_n(\vec{x}_n)$ be the actions available to a robot, then this is the corresponding closure axiom:

$$(\forall a)\,(\exists \vec{x}_1)\,a = A_1(\vec{x}_1) \vee \ldots \vee (\exists \vec{x}_n)\,a = A_n(\vec{x}_n)$$

Example 4 Recall robot *Blindie* from the introduction, who may be in front of an open door without knowing the state of that door:

$$\neg Holds(Closed(Door1), S_0) \quad (20)$$
$$Knows(InFrontOf(Door1), S_0) \quad (21)$$
$$\neg Knows(\neg Closed(Door1), S_0) \quad (22)$$

The goal of entering the room cannot be achieved by this robot without assistance. Although it could move into the room through the open door, the robot has no way of arriving at this conclusion if it is not able to sense the states of doors: Let Ax denote the conjunction of the axioms just mentioned, the precondition and the state and knowledge update axioms for $Press(Button(x))$ from above, the foundational axioms including the induction and accompanying axioms, and the following closure axiom, stating that pressing buttons is the only action available to the robot:

$$(\forall a)\,(\exists x)\,a = Press(Button(x))$$

Then the robot can never know whether *Door1* is open or not:

$$Ax \models (\forall s)\,(S_0 \leq s \supset \neg Knows(Closed(Door1), s) \wedge \neg Knows(\neg Closed(Door1), s)$$

This follows from the induction axiom instantiated by $\{\Pi/\lambda s.\, F\}$ where F denotes the entire formula in the range of the quantification. The base case, S_0, is given by (22) and by (20) in conjunction with (11). For the induction step, consider the only action $a = Press(Button(x))$ in conjunction with knowledge update axiom (14) and foundational axiom (12). Then Ax entails,

$$Poss(Press(Button(x)), s) \supset$$
$$x \neq Door1 \supset$$
$$(\forall z)\,(KState(Do(Press(Button(x)), s), z) \supset$$
$$[\,Holds(Closed(Door1), z) \equiv$$
$$(\exists z')\,KState(s, z') \wedge Holds(Closed(Door1), z')\,])$$

and

$$Poss(Press(Button(x)), s) \supset$$
$$x = Door1 \supset$$
$$(\forall z)\,(KState(Do(Press(Button(x)), s), z) \supset$$
$$[\,Holds(Closed(Door1), z) \equiv$$
$$(\exists z')\,KState(s, z') \wedge \neg Holds(Closed(Door1), z')\,])$$

The induction step now follows from the induction hypothesis that the robot does not know the status of *Door1* in s. ∎

Example 5 Recall robot *Dumbie* from the introduction, who is aware of the fact that a door is somehow under the control of a button next to it without knowing the precise causal relation, namely, that pressing the button always alters the state of the door. Hence, while the state update itself is still suitably described by the two axioms of (10), this is the knowledge update axiom characterizing *Dumbie*:

$$\begin{array}{l} Poss(Press(Button(x)), s) \supset \\ \quad (\forall z)\,(KState(Do(Press(Button(x)), s), z) \equiv \\ \qquad (\exists z')\,(KState(s, z') \wedge \\ [Holds(Closed(x), z') \supset z = z' \vee z \circ Closed(x) = z'] \wedge \\ [\neg Holds(Closed(x), z') \supset z = z' \vee z = z' \circ Closed(x)])) \end{array}$$

(Compare this to knowledge update axiom (14), which encodes accurate effect knowledge.) Although it could open a closed door by pressing the button next to it, the robot does not know this: Let Ax denote the knowledge update axiom just mentioned along with the precondition and state update axioms for $Press(Button(x))$ from above, the foundational axioms including including the induction and accompanying axioms, the axiom

$$Knows(Closed(Door1), S_0) \tag{23}$$

and the closure axiom

$$(\forall a)\,(\exists x)\,a = Press(Button(x))$$

Then the robot can never know that it is possible to open the door:

$$Ax \models (\forall s)\,(S_0 \leq s \supset \neg Knows(\neg Closed(Door1), s)$$

This follows from the induction axiom instantiated by $\{\Pi/\lambda s.\, F\}$ where F denotes the entire formula in the range of the quantification. The base case, S_0, is given by (23) in conjunction with (11). For the induction step, consider the only action $a = Press(Button(x))$ in conjunction with the knowledge update axiom of *Dumbie* for this action and foundational axiom (12). Then Ax entails,

$$\begin{array}{l} Poss(Press(Button(x)), s) \supset \\ \quad x \neq Door1 \supset \\ \qquad (\forall z)\,(KState(Do(Press(Button(x)), s), z) \supset \\ \qquad\quad [\,Holds(Closed(Door1), z) \equiv \\ (\exists z')\,KState(s, z') \wedge Holds(Closed(Door1), z')\,]) \end{array}$$

and

$$\begin{array}{l} Poss(Press(Button(x)), s) \supset \\ \quad x = Door1 \supset \\ \qquad (\exists z)\,(KState(Do(Press(Button(x)), s), z) \supset \\ \qquad\qquad Holds(Closed(Door1), z)) \end{array}$$

The induction step now follows from the induction hypothesis. ∎

7 DISCUSSION

We have developed a formal account of a robot's changing knowledge about the state of its environment. Based on the established predicate calculus formalism for reasoning about actions of the Fluent Calculus, our approach is kept representationally and inferentially simple in that it avoids non-classical extensions to standard predicate logic. Our formalism accounts for both knowledge preconditions of actions and information gathering actions which enhance the state knowledge of a robot. Our theory also provides simple and elegant means to reason about what a robot does *not* know and about goal achievability.

The effect of actions on state knowledge is specified by so-called knowledge update axioms, by which is solved the representational Frame Problem for knowledge according to the main theorem of the Fluent Calculus for sensing (Item 2 of Theorem 4). Moreover, knowledge update axioms have been shown to lay the foundations for overcoming the inferential aspect of this Frame Problem, too.

We have axiomatically introduced the concept of a conditional action, by which a robot may condition its intended course of actions on the result of a sensing action included in its plan. Our *If*-construct uses only atomic conditions and only allows for a single conditionally executed action. The integration of more expressive notions, say arbitrarily complex conditions and unbounded iteration [Levesque, 1996], is considered an important direction of future research. Following [Levesque *et al.*, 1997], such complex actions can be dealt with in two fundamentally different ways: They can be defined via macros or be integrated into the language. The former approach, actually taken in [Levesque *et al.*, 1997], has the drawback that non-sequential plans (that is, which include conditional and loop statements) cannot be planned by deduction as they are not part of the language. The latter approach, on the other hand, requires complete reification of arbitrary formulas. Inasmuch as reification plays an important role in the simple Fluent Calculus anyway, it is this second alternative approach which seems most promising a route to take towards an extension of our formalism.

Knowledge and sensing actions were first investigated in [Moore, 1985] in the context of the Situation Calculus [McCarthy, 1963], and in [Scherl and Levesque, 1993] this approach was combined with the solution

to the Frame Problem provided by so-called successor state axioms [Reiter, 1991]. The basic idea of this approach is to represent state knowledge by a binary situation-situation relation $K(s, s')$, meaning that as far as the robot knows in situation s it could as well be in situation s'. Hence, every given fact about any such s' is considered possible by the robot. Having readily available the explicit notion of a state in the Fluent Calculus, our formalization avoids this indirect encoding of state knowledge, which is intuitively less appealing because it seems that a robot should always know exactly which *situation* it is in—after all, situations in the Situation Calculus are merely sequences of actions that have been or will be taken by the robot [Levesque *et al.*, 1998].

Apart from this clash of intuitions, there is a more crucial difference between our approach and that of [Moore, 1985; Scherl and Levesque, 1993]: The latter defines the effect of a non-sensing action a on the robot's state knowledge via the equivalence relation $K(Do(a, s), s'') \equiv (\exists s')(K(s, s') \wedge s'' = Do(a, s'))$. Hence, the very same successor state axioms apply to both the state update (when moving from s to $Do(a, s)$) and the knowledge update (when moving from s' to $s'' = Do(a, s')$). In contrast, with independent specifications of state and knowledge update, our formalism furnishes a ready approach for representing and reasoning about goal achievability that is possibly restricted due to limited knowledge of the effects of actions. This separating what a user knows from what a robot knows distinguishes our theory from other existing accounts of sensing action and knowledge, too, such as [Lobo *et al.*, 1997; Son and Baral, 1998], where also non-sensing actions have identical effect on the external and internal states.

Representing and reasoning about non-knowledge has previously been realized in the context of the Situation Calculus [Lakemeyer and Levesque, 1998; Lakemeyer and Levesque, 1999]. Two approaches to 'only knowing' have been offered, one of which is by a non-standard semantics while the other one is an axiomatization in classical logic but with two complex second-order axioms involved. Exploiting the reification of fluents and states, knowledge and non-knowledge can be expressed in our approach by mere first-order sentences along with the standard semantics of classical predicate logic.

Goal achievability has been analyzed previously in [Lin and Levesque, 1997], independently of the second author's approach to knowledge and sensing. The result of the present paper can thus be viewed as a unifying theory for representing and reasoning about knowledge, sensing, and mental ability of achieving goals.

Our further plans for future work include the integration of the formalism for reasoning about a robot's knowledge into existing extensions of the basic Fluent Calculus. A particularly interesting combination is that of ramifications and knowledge. Just like it may have only restricted knowledge of the direct effects of actions, a robot may lack knowledge of state constraints and of indirect effects. The solution to the Ramification Problem of [Thielscher, 1997] is readily available for a combination with knowledge update axioms to allow a robot to reason about the indirect effects it is aware of.

References

[Bibel, 1998] Wolfgang Bibel. Let's plan it deductively! *Artificial Intelligence*, 103(1–2):183–208, 1998.

[Hölldobler and Schneeberger, 1990] Steffen Hölldobler and Josef Schneeberger. A new deductive approach to planning. *New Generation Computing*, 8:225–244, 1990.

[Hölldobler and Thielscher, 1995] Steffen Hölldobler and Michael Thielscher. Computing change and specificity with equational logic programs. *Annals of Mathematics and Artificial Intelligence*, 14(1):99–133, 1995.

[Lakemeyer and Levesque, 1998] Gerhard Lakemeyer and Hector J. Levesque. $\mathcal{AOL}$: A logic of acting, sensing, knowing, and only knowing. In A. G. Cohn, L. K. Schubert, and S. C. Shapiro, editors, *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 316–327, Trento, Italy, 1998.

[Lakemeyer and Levesque, 1999] Gerhard Lakemeyer and Hector J. Levesque. Query evaluation and progression in $\mathcal{AOL}$ knowledge bases. In T. Dean, editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 124–131, Stockholm, Sweden, 1999.

[Levesque *et al.*, 1997] Hector J. Levesque, Raymond Reiter, Yves Lespérance, Fangzhen Lin, and Richard B. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1–3):59–83, 1997.

[Levesque *et al.*, 1998] Hector Levesque, Fiora Pirri, and Ray Reiter. Foundations for a calculus of

situations. *Linköping Electronic Articles in Computer and Information Science*, 3(18), 1998. URL: http://www.ep.liu.se/ea/cis/1998/018/.

[Levesque, 1996] Hector J. Levesque. What is planning in the presence of sensing? In B. Clancey and D. Weld, editors, *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 1139–1146, Portland, OR, August 1996. MIT Press.

[Lin and Levesque, 1997] Fangzhen Lin and Hector Levesque. What robots can do: Robot programs and effective achievability, 1997. (Manuscript).

[Lobo *et al.*, 1997] Jorge Lobo, Gisela Mendez, and Stuart R. Taylor. Adding knowledge to the action description language $\mathcal{A}$. In B. Kuipers and B. Webber, editors, *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 454–459, Providence, RI, July 1997. MIT Press.

[McCarthy, 1963] John McCarthy. *Situations and Actions and Causal Laws.* Stanford Artificial Intelligence Project, Memo 2, 1963.

[Moore, 1985] Robert Moore. A formal theory of knowledge and action. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358. Ablex, 1985.

[Reiter, 1991] Ray Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation*, pages 359–380. Academic Press, 1991.

[Reiter, 1993] Ray Reiter. Proving properties of states in the situation calculus. *Artificial Intelligence*, 64:337–351, 1993.

[Scherl and Levesque, 1993] Richard Scherl and Hector Levesque. The frame problem and knowledge-producing actions. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 689–695, Washington, DC, July 1993.

[Son and Baral, 1998] Tran Cao Son and Chitta Baral. Formalizing sensing actions—a transition function based approach, 1998. (Manuscript).

[Störr and Thielscher, 2000] Hans-Peter Störr and Michael Thielscher. A new equational foundation for the fluent calculus, 2000. (Manuscript.) URL: http://pikas.inf.tu-dresden.de/~mit/publications/conferences/FCeq.ps.

[Thielscher, 1997] Michael Thielscher. Ramification and causality. *Artificial Intelligence*, 89(1–2):317–364, 1997.

[Thielscher, 1998] Michael Thielscher. Introduction to the Fluent Calculus. *Electronic Transactions on Artificial Intelligence*, 2(3–4):179–192, 1998. URL: http://www.ep.liu.se/ea/cis/1998/014/.

[Thielscher, 1999] Michael Thielscher. From Situation Calculus to Fluent Calculus: State update axioms as a solution to the inferential frame problem. *Artificial Intelligence*, 111(1–2):277–299, 1999.

A Logic Programming Approach to Conflict Resolution in Policy Management

Jan Chomicki*
Monmouth University
chomicki@monmouth.edu

Jorge Lobo, Shamim Naqvi
Bell Labs
{jlobo,shamim}@research.bell-labs.com

Abstract

The simple *event-condition-action* (ECA) rule paradigm of active databases has proved very useful in many AI and database applications. However, its applicability goes beyond data management. ECA rules can be used in network management and monitoring, electronic commerce, security and access management, and other application areas to express *policies* – collections of general principles specifying the desired behavior of a system. In this paper we use a declarative policy description language $\mathcal{PDL}$, in which policies are formulated as sets of ECA rules. The main contribution of the paper is a framework for detecting action conflicts and finding resolutions to these conflicts. Conflicts are captured as violations of action constraints. The semantics of rules, and conflict detection and resolution are defined axiomatically using logic programs. Given a policy and a set of action constraints the framework defines a monitor that filters the output of the policy to satisfy the constraints. We briefly describe the architecture of a $\mathcal{PDL}$-based policy server being used to provide centralized administration of a soft switch in a communication network and show how it can be augmented to handle conflict resolution.

1 Introduction

The simple *event-condition-action* rule paradigm has proved very useful in many AI and database applications [23, 4], from constraint maintenance to the general encoding of expert rules. However, the applicability of the *event-condition-action* rule paradigm goes beyond data management or expert systems. Such rules can be used in network management and monitoring [11], electronic commerce [9], security and access management [17], and other application areas to express *policies* – collections of general principles specifying the desired behavior of a system. For instance, network management is mainly carried out by following policies about the behavior of the resources in the network. The policies are usually formulated as sets of low-level rules that describe how to (re)configure a device or how to manipulate the different network elements under different network conditions. Analogous policies occur in areas such as electronic commerce ("orders from established customers should receive higher priority") and computer security. Usually, policies are coded in an imperative programming language like Java. This makes for implementation ease and efficiency but limits what can be done with policies. For instance, it is difficult to modify, verify or analyze such policies. In this paper, we pursue a different approach. We use a declarative *policy description language* $\mathcal{PDL}$ [15], in which policies are formulated as sets of *event-condition-action* rules of the form

event **causes** *action* **if** *condition*. (1)

A policy rule can be read as: if the *event* occurs in a situation where the *condition* is true, then the *action* is executed. A policy in $\mathcal{PDL}$ defines a *transducer:* a function that maps sets of events into sets of actions. The $\mathcal{PDL}$ policy manager provides an implementation for such transducers.

A formal description of the syntax and semantics of $\mathcal{PDL}$ can be found in Section 2. However, our interest in this paper is not in the language per se but in controlling policies written in the language. In particular, we address the issues of *conflict detection* and *resolution*.

Example 1 Consider the following simple scenario:

*Work done while visiting Bell Labs.

There is an automatic reservation system for the conference room in a department. The reservations are controlled by a simple first-in-first-served policy captured by the following rule:

$$requestRes(User) \textbf{ causes } processRes(User). \quad (2)$$

The procedure *processRes* reserves the room for the user if it is available. Notice, however, that if there are simultaneous reservation requests from different users, there will be two instances of the rule triggered and the policy will not work unless one of the *processRes* calls is cancelled. We introduce *action constraints* into $\mathcal{PDL}$ to capture this type of conflicts. The constraints describe under which circumstances a set of actions cannot be executed simultaneously. For the reservation policy the appropriate action constraint is

$$\textbf{never } processRes(U_1) \wedge processRes(U_2) \textbf{ if } U_1 \neq U_2.$$

Typically the burden of conflict resolution is left in the hands of the policy administrator, who must provide the resolution of the conflicts in the code that implements the policies. The process is *ad hoc* with no guarantees on the properties of the solution.

This paper introduces a formal framework for detecting rule conflicts and finding resolutions to these conflicts. Given a policy and a set of constraints on the concurrent execution of actions, the framework produces a monitor for the policy. A policy monitor filters the output of a policy (i.e., a set of actions) by cancelling some actions to obtain a result consistent with the constraints. However, actions cannot be cancelled arbitrarily. First, we would like to cancel as few actions as possible. Second, we propose that policy monitors should be *unobtrusive:* For any set of input events E, an unobtrusive monitor outputs the same set of actions as the original policy applied to a *subset* E' of E. Thus, if an event e is in E', every action caused by e succeeds. If an event e is in $E \setminus E'$, every action caused by e fails unless it is also caused by a different event which is in E'. This is analogous to the *atomicity* requirement for standard database transactions. In our context, it means that the execution of an unobtrusive monitor corresponds to *some* conflict-free execution of the original policy.

In this paper, the semantics of policy rules, and conflict detection and resolution are defined axiomatically using disjunctive logic programs. Such programs are generated automatically from $\mathcal{PDL}$ specifications and can subsequently be modified by the user. The latter option allows for more flexibility in handling conflicts. The logic programming formulation has many desirable properties: It is both executable (using well-established logic programming techniques) and easy to modify and analyze formally.

The plan of the paper is as follows. In Section 2, we define the syntax and semantics of $\mathcal{PDL}$ (the latter using a translation to Horn rules). In Section 3, which constitutes the core of the paper, we address the issue of conflict detection and resolution using a logic programming framework. The construction described deals with unobtrusive monitors. In this section we restrict the monitors to policies without negative events (i.e. a policy cannot be based on the absence of an event). In Section 4, we study relevant computational complexity issues. In Section 5, we discuss the problems of having negated events and present a solution. In Section 6, we show how the unobtrusiveness requirement can be partially relaxed using an elaboration of our basic framework. In Section 7, we outline the existing implementation of a $\mathcal{PDL}$ server and show how it can be augmented to support policy monitors. In Section 8, we survey related work. In Section 9, we conclude and outline the directions for further research. Due to space limitations the examples used are simplified and only selected proofs are given.

2 The policy description language $\mathcal{PDL}$

2.1 Syntax

The language we consider consists of three basic classes of symbols: *primitive event* symbols, *action* symbols and *constant* symbols. These symbols are system-dependent and are given to the user that defines the policies. There is also a set of standard ordered types such as integers, floats, character strings, etc. Action and primitive event symbols may be of any nonnegative arity. Each action symbol of arity n denotes the name of a procedure that takes n arguments (also called parameters) of a particular type. Every event argument and constant symbol also has an associated type. The arguments of event symbols represent event attributes.

Definition 1 A *policy* is a finite collection of well-typed policy rules of the form (1), where the *event*, *action* and *condition* parts of a rule are defined below.

Definition 2 The *event* part of a policy rule is an expression of the form $e_1 \& \ldots \& e_n$ where each e_i is an *event literal.* An *event literal* is either a typed *event term* of the form $e(t_1, \ldots, t_n)$, where e is a primitive event symbol of n arguments and each t_i is a constant or a variable, or a primitive event symbol e preceeded

by ! (!e) representing the negation of e. An *event instance* is a ground event term.

Definition 3 The *action* part of a policy rule is a typed *action term* of the form $a(t_1, \ldots, t_n)$, where a is an action symbol of n arguments and each t_i is a variable that appears in the event part of the rule or a constant.

Definition 4 The *condition* part of a policy rule is an expression of the form $p_1, \ldots, p_n$, where each p_i is a predicate of the form $t_1 \theta t_2$, θ is a relation operator from the set $\{=, \neq, <, \leq, >, \geq\}$ and each t_i is either a variable that appears in the event part or a constant. The condition represents the conjunction of the predicates.

2.2 Semantics

We adopt the view that a policy expects as an input a set of instances of primitive events assuming the events in the set occurred simultaneously. (The granularity of time is application dependent: It may be a minute in some contexts, a day - in others. We assume that the selection of the time granularity is made outside of our policy framework.)

Definition 5 A finite set of event instances is an *epoch*. The set of all possible epochs is denoted by *Epochs*. We say that the event e *occurs* in an epoch if an instance of the event term $e(X_1, \ldots, X_n)$ is a member of the epoch. We say that the event literal !e *occurs* in an epoch (with a single instance) if there are no instances of the primitive event e in the epoch. The set of all ground action terms is denoted by *Actions* and its finite subsets are called *action sets*.

In the following, assume P is a policy. Formally, the semantics of P is given by a transducer $\pi_P : Epochs \longrightarrow 2^{Actions}$. We define this transducer using a Horn logic program Π_P of a special form. The minimal model of this program, which can be computed using well-known logic programming techniques, represents the transducer defined by the policy. To compute the actions that are triggered by an epoch, we transform the epoch into a set of ground atoms and add them to the program Π_P. The actions will appear in the minimal model of the expanded program. For an epoch E, let

$$occ(E) = \{occ(g) | g \in E\} \cup \{occ(!e) | e \text{ has no instances in } E\}.$$

The intuitive meaning of $occ(e(t_1, \ldots, t_n))$ is that the instance $e(t_1, \ldots, t_n)$ of the primitive event e *occurred* in the current epoch. Then, each rule of the form

$$e_1 \& \ldots \& e_l \textbf{ causes } a \textbf{ if } C$$

in the policy is translated into the implication

$$exec(a) \leftarrow occ(e_1) \wedge \ldots \wedge occ(e_l) \wedge C.$$

We denote by Π_P the set of rules that result from the translation of each policy rule in P. It is easy to see that Π_P is a non-recursive Horn logic program.

Definition 6 $\pi_P : Epochs \longrightarrow 2^{Actions}$ is the transducer defined by P if for every epoch E, $a \in \pi_P(E)$ iff $\Pi_P \cup occ(E) \models exec(a)$.

By the virtue of Π_P being a non-recursive Horn program, the evaluation of π_P can be done in time linear with respect to the size of the input epoch.

3 Conflict detection and resolution

3.1 Action constraints and monitors

A policy generates a conflict when its output contains a set of actions that the policy administrator has specified *cannot occur together*. The conflicts are captured as violations of action constraints.

Definition 7 An *action constraint* is an expression of the form

$$\textbf{never } a_1 \wedge \ldots \wedge a_m \textbf{ if } C.$$

Each a_i is an action term and C is a condition such that variables appearing in C also appear in one of the a_is. The meaning is: *"never allow the simultaneous execution of the actions $a_1, \ldots, a_m$ if the condition C holds."* It formally represents the formula

$$\forall \neg (a_1 \wedge \ldots \wedge a_n \wedge C).$$

Definition 8 Given an action set S consisting of ground action terms, we say that S *satisfies* an action constraint ac (resp. a set of action constraints AC) if S is a model of ac (resp. of all the constraints in AC) in the standard model theoretic sense (with action terms viewed as literals).

A policy monitor of a set of action constraints generates only action sets without conflicts, i.e., satisfying all given action constraints.

Definition 9 Given a set of action constraints AC, a *policy monitor* ω_{AC} is a transducer $\omega_{AC} : Epochs \longrightarrow 2^{Actions}$ such that for every epoch E, $\omega_{AC}(E)$ satisfies AC.

Note that a monitor, being a transducer, is semantically identical to a policy. However, unlike policies,

monitors will not be defined in $\mathcal{PDL}$ but rather specified indirectly using logic programs of a special form (defined later in this section).

Next, we identify several important properties of policy monitors. They capture the intuition that monitors should be chosen to behave as close as possible to the policy which output they filter and that the effect of conflict resolution should be maximally transparent to the user.

In the following definitions, assume that P is a policy and AC is a set of action constraints. They will be omitted when they are clear from the context.

Definition 10 An epoch E is *P-consistent* with AC if $\pi_P(E)$ satisfies AC. A *(P, AC)-consistent reduction* E' of an epoch is an epoch such that $E' \subseteq E$ and E' is P-consistent with AC. The reduction is *maximal* if there is no (P, AC)-consistent reduction E'' of E such that $E' \subseteq E''$ and $E' \neq E''$.

Definition 11 A policy monitor ω_{AC} is:

1. a *conservative* monitor of P if it is identical to the policy for P-consistent epochs.

2. an *action-cancellation* monitor of P if does not generate any actions beyond those of P, i.e., for every epoch E, $\omega_{AC}(E) \subseteq P(E)$.

3. a *maximal action-cancellation* monitor of P if for every epoch E, $\omega_{AC}(E)$ is a maximal subset of $P(E)$ that satisfies AC.

4. an *event-cancellation* monitor of P if for every epoch E, there exists a (P, AC)-consistent reduction E' of E such that $\omega_{AC}(E) = P(E')$.

5. a *maximal event-cancellation* monitor of P if for every epoch E, there exists a maximal (P, AC)-consistent reduction E' of E such that $\omega_{AC}(E) = P(E')$.

Conservativeness is a basic requirement that all monitors should satisfy. All the monitors described in this paper work by cancelling actions to eliminate conflicts. It is easy to see that in the absence of negated events every event-cancellation monitor is also an action-cancellation monitor. However, there is a difference between action- and event-cancellation monitors. The latter satisfy additionally a property that we call *unobtrusiveness*. To appreciate the importance of unobtrusiveness, consider the following example.

Example 2 Assume we have the policy P_1

defectiveProduct **causes** *stop*
orderReceived **causes** *mailProduct*
orderReceived **causes** *chargeCC*

and the constraint

never *stop* $\wedge$ *mailProduct*.

If the events *defectiveProduct* and *orderReceived* occur together, the resulting conflict may be eliminated by cancelling the action *mailProduct*. However, the action *chargeCC* can still be executed without conflict, although this is intuitively incorrect (at least from the customer's point of view!). An event-cancellation monitor avoids this problem by ignoring the event *orderReceived* and therefore also indirectly cancelling both actions it causes.

Example 3 To see the need for action-cancellation monitors which are not unobtrusive, consider the rule

request(X) **causes** *acknowledge*(X).

If there are other rules involving the *request* event that lead to conflict and as the result some other actions caused by this event are cancelled, the above rule should still be executed. Thus, the *request* event cannot simply be ignored.

The next question to ask is if we can forgo event-cancellation and only work with action-cancellation monitors. If we go back to Example 2, at first glance it seems that we could make an explicit connection between *mailProduct* and *chargeCC* to solve the problem. One possibility would be to have rules with multiple actions such as

orderReceived **causes** *mailProduct*, *chargeCC*

and when one of the two actions is cancelled the other should also be cancelled. A similar effect can be achieved by adding the constraint

never *stop* $\wedge$ *chargeCC*.

However, both of those approaches may unnecessarily or even incorrectly cause the cancellation of an action. This may occur in the example above if there is another event also causing *chargeCC*. We could, for example, have the rule

serviceCallCompleted **causes** *chargeCC*.

If the event *serviceCallCompleted* occurs simultaneously with *defectiveProduct* and *orderReceived*, an

action-cancellation monitor will not charge for the service call.

A comprehensive policy management system should provide both unobtrusive and non-unobtrusive monitors, as well as their combinations. In this paper, we concentrate first on *event-cancellation monitors* which are unobtrusive. Afterwards we show how to combine unobtrusive and not unobtrusive monitors for different parts of a policy.

3.2 Translation to logic programs

Here we define policy monitors for policies that do not involve negated events by extending the Datalog translation of policies, defined in the previous section. The idea is to capture conflict resolution *at the level of rule execution*. In addition to the two predicate symbols *occ* and *exec*, we introduce three new predicate symbols: *block* (an action), *ignore* (an event), and *accept* (an action).

Conflict rules Ψ_{AC}. Each constraint in AC of the form

$$\textbf{never } a_1 \wedge \ldots \wedge a_n \textbf{ if } C$$

is translated into a *conflict* rule

$$block(a_1) \vee \ldots \vee block(a_n) \leftarrow exec(a_1) \wedge \ldots \wedge exec(a_n) \wedge C.$$

Blocking rules B_P and accepting rules A_P. Each policy rule in P of the form

$$e_1 \& \ldots \& e_n \textbf{ causes } a \textbf{ if } C$$

is translated into a *blocking* rule

$$ignore(e_1) \vee \ldots \vee ignore(e_n) \leftarrow occ(e_1) \wedge \ldots \wedge occ(e_n) \wedge C \wedge block(a)$$

and an *accepting* rule

$$accept(a) \leftarrow occ(e_1) \wedge \ldots \wedge occ(e_n) \wedge C \wedge \neg ignore(e_1) \wedge \ldots \wedge \neg ignore(e_n).$$

Example 4 The constraint

$$\textbf{never } stop \wedge mailProduct$$

is translated to

$$block(stop) \vee block(mailProduct) \leftarrow exec(stop) \wedge exec(mailProduct).$$

Example 5 The constraint

$$\textbf{never } processRes(U_1) \wedge processRes(U_2) \textbf{ if } U_1 \neq U_2$$

is translated to

$$block(processRes(U_1)) \vee block(processRes(U_2)) \leftarrow exec(processRes(U_1)) \wedge exec(processRes(U_2)) \wedge U_1 \neq U_2.$$

Example 6 The rule

$$dial \ \& \ charge \textbf{ causes } connect$$

is translated to

$$ignore(dial) \vee ignore(charge) \leftarrow occ(dial) \wedge occ(charge) \wedge block(connect)$$
$$accept(connect) \leftarrow occ(dial) \wedge occ(charge) \wedge \neg ignore(dial) \wedge \neg ignore(charge).$$

Remarks: Note that all the rules above are *safe*. That is, variables that appear in the consequents of the implications or in negated literals in the antecedent also appear in literals with no negation in the antecedents. The rules are not, strictly speaking, in Datalog, since actions and events are encoded as function symbols. However, those symbols are not nested and thus the resulting program can easily be translated into Datalog by introducing new predicate symbols. Also, for a policy P and a set of constraints AC, the program $\Pi_P \cup \Psi_{AC} \cup A_P \cup B_P$ is a *hierarchical (i.e. non-recursive) disjunctive logic program.*

Theorem 1 For any epoch E, E' is a maximally (P, AC)-consistent reduction of E iff $E' = \{e | occ(e) \in M \wedge ignore(e) \notin M\}$ for some minimal model M of $\Pi_P \cup \Psi_{AC} \cup A_P \cup B_P \cup occ(E)$.

This theorem provides us with sufficient information to build a maximal event-cancellation monitor. We only need any of the standard algorithms to compute the minimal models [21] of disjunctive logic programs.

Definition 12 Let $\Omega_{P,AC} : Epochs \longrightarrow 2^{Actions}$ be a transducer defined for any epoch E in the following way:

1. Select a minimal model M of $\Pi_P \cup \Psi_{AC} \cup A_P \cup B_P \cup occ(E)$.
2. Return the set $\{a | accept(a) \in M\}$ as output (i.e. $\Omega_{P,AC}(E) = \{a | accept(a) \in M\}$).

Corollary 1 $\Omega_{P,AC}$ is a maximal event-cancellation monitor of P. Moreover, all maximal event-cancellation monitors of P can be obtained in this way.

It is also immediate to see that $\Omega_{P,AC}$ is conservative (if no conflicts occur, no actions are blocked and no events ignored).

3.3 Tuning the translation

The logic program translation defined above encodes all maximal event-cancellation monitors that can be associated with a policy. However, in real situations there are monitors that can be considered more appropriate than others. When there is a choice of blocking one of several actions to resolve a conflict, the application domain may suggest a *priority ordering* among the actions.

Example 7 Let's return to Example 2. The constraint says that we can never execute simultaneously the *stop* and *mailProduct* actions. Naturally, if the product is defective we should block *mailProduct* and let *stop* proceed. In other words, we give *priority* to *stop* over *mailProduct*. Priority can be encoded by changing the conflict rule to

$$block(mailProduct) \leftarrow exec(stop) \wedge exec(mailProduct)$$

eliminating the disjunction. Thus, for each constraint the user can select the action with the least priority and remove all the others from the head of the corresponding conflict rule. User choices for different constraints need to be coordinated to make sure the resulting monitor is still maximal. How to do it is beyond the scope of this paper.

A similar situation arises when choosing events to ignore, since in many cases it is impossible, incorrect, or undesirable to ignore certain events, e.g., time events. We call such events *persistent*.

Example 8 In Example 6 the *dial* event is persistent. Therefore, the blocking rule should be changed to

$$ignore(charge) \leftarrow occ(dial) \wedge occ(charge) \wedge block(connect).$$

In general, users can selectively remove ignored events from the consequent of the blocking rules to make them persistent until at least one is left.

4 Computational complexity

Assuming a class $\mathcal{C}$ of policy monitors, a fixed $\mathcal{PDL}$ policy P and a fixed set AC of action constraints, we study the complexity of simulating monitors from $\mathcal{C}$ as a function of the size of a given input epoch E and a given set of actions A. The simulation consists of determining whether $A \subseteq \omega_{AC}(E)$ for some monitor ω_{AC} from $\mathcal{C}$.

Theorem 2 Simulating maximal action-cancellation monitors can be done in PTIME.

Proof: The simulation returns *yes* iff $A \subseteq P(E)$ and A satifies AC. If this is the case, one can obtain in PTIME the action set $\omega_{AC}(E)$ for some maximal action-cancellation monitor ω_{AC} using a greedy approach: Start with A and keep adding actions in $P(E) - A$ until you obtain a set that cannot be further expanded without violating the constraints AC. □

Theorem 3 Simulating maximal event-cancellation monitors is NP-complete.

Proof:

Upper bound. From the general results about the complexity of minimal models for Disjunctive DATALOG programs, it follows that the simulation problem is in Σ_2^p [6]. This bound can be sharpened to NP as follows. First, note that as soon as the set of events to be ignored is determined, the set of accepted actions can be computed by a single application of accepting rules and thus in PTIME. Second, for a given epoch the set S of *occ* and *exec* atoms can be computed in linear time. Once this is done, we take the ground instantiation of the entire logic program and keep only those rules whose all *occ* and *exec* atoms in the body are in S and (ground) conditions are true. For those rules the atoms in S and the conditions are erased. We are left thus with two kinds of clauses:

$$b_1 \vee \cdots \vee b_n$$

and

$$i_1 \vee \cdots \vee i_m \vee \neg b_0$$

where b_i are *block* atoms and i_j are *ignore* atoms. Moreover, for a fixed policy and a fixed set of action constraints the number of literals in those clauses is bounded. By exhaustively resolving the clauses of the first kind with those of the second, we obtain an equivalent set S^+ of polynomial size, containing only positive clauses. For any guessed set I of *ignore* atoms it can be verified in PTIME if it forms, together perhaps with some *block* atoms, a minimal model of S^+. Similarly, it can be checked in PTIME whether, given I, the set of accepted actions contains the given set of actions A.

Lower bound. Reduction from 3-COLORABILITY. We need four unary event symbols (for the different colors): r, b and g and one binary event symbol *arc*.

The policy P_0 is defined as follows:

$r(X)\&b(X)$ **causes** $fail$
$r(X)\&g(X)$ **causes** $fail$
$b(X)\&g(X)$ **causes** $fail$
$arc(X,Y)\&r(X)\&r(Y)$ **causes** $fail$
$arc(X,Y)\&b(X)\&b(Y)$ **causes** $fail$
$arc(X,Y)\&g(X)\&g(Y)$ **causes** $fail$
$r(X)$ **causes** $ok(X)$
$b(X)$ **causes** $ok(X)$
$g(X)$ **causes** $ok(X)$
$arc(X,Y)$ **causes** $ok\_arc(X,Y)$.

The set of action constraints AC_0 consists of the single action constraint

never $fail$.

A graph $G = (V,E)$ is 3-colorable iff there exists a maximal (P_0, AC_0)-consistent reduction E' of the epoch

$$\{r(v)|v \in V\} \cup \{b(v)|v \in V\} \cup \{g(v)|v \in V\}$$
$$\cup\{arc(u,v)|(u,v) \in E\}$$

such that $P_0(E')$ contains

$$\{ok(v)|v \in V\} \cup \{ok\_arc(u,v)|(u,v) \in E\}.$$

□

We would like to remark that if the set of actions A is fixed, it can be shown that the simulation problem for event-cancellation monitors is in PTIME.

Tuning the translation of $\mathcal{PDL}$ policy monitors, as described above, reduces the number of disjuncts in clause heads. If we are left with a (non-recursive) logic program, the simulation problem is in LOGSPACE.

5 Negated events

There are several problems with extending our approach to policies with negated events. First, in the presence of negated events, policies may not have *any* policy monitors.

Example 9 Consider the following policy P_3:

e_1 **causes** a_0
$!e_1$ **causes** a_1
e_1 **causes** a_2
$!e_1$ **causes** a_3

and the constraints $\{$**never** $a_0 \wedge a_2$, **never** $a_1 \wedge a_3\}$. If e_1 occurs there is a conflict. If it is ignored there is also a conflict.

Even more complications arise if the events have attributes since two instances of the same event can occur simultaneously and ignoring one instance does not imply ignoring the second instance. In general, testing consistency is computationally difficult.

Theorem 4 Determining the existence of an event-cancellation monitor for a given set of $\mathcal{PDL}$ rules (where events have no attributes) and a given epoch is co-NP-complete.

The second problem is due to the *nonmonotonic* character of negation. Our monitors work by action cancellation. In the presence of negation, cancelling an action and ignoring an event causing it may trigger some new actions.

Example 10 The following policy P_4:

$!e_1$ **causes** a_1
e_1 **causes** a_2
e_1 **causes** a_3

and the constraint $\{$**never** $a_2 \wedge a_3\}$ has a maximal event-cancellation monitor: one that outputs a_1 for every epoch. However, this monitor is not an action-cancellation monitor because $a_1 \notin \pi_{P_4}(\{e_1\}) = \{a_2, a_3\}$. Moreover, the logic programming translation specifies a different monitor: one that returns an empty set of actions for every epoch. This monitor is no longer maximal.

To deal with negative events, we are thus forced to loosen the definition of an epoch and let some events be undefined (i.e., neither the event nor its negation are known to have occurred).

Definition 13 An *extended epoch* is a collection of instances of events plus a subset of $\{!e : e$ is a primitive event symbol$\}$. An extended epoch E is *coherent* if for every negated event $!e$ in E there is no instance of e in E; *complete* if for every primitive event e that has no instances in the epoch, $!e$ is a member of E.

Under this definition, there are three possible coherent epochs in Example 9: $\{\}$, $\{e_1\}$ and $\{!e_1\}$. This definition naturally induces a new definition of *extended policies*. The input of an extended policy is an extended epoch and the output a set of actions. If we apply the policy in Example 9 to the extended empty epoch the policy outputs no actions. We can also define *extended conservative* monitors, regular and maximal *extended action-cancellation* monitors, and regular and maximal *extended event-cancellation* monitors with the appropriate modifications in the original definitions.

Note that the existence of the empty epoch guarantees that there is always an extended action-cancellation monitor, namely the extended monitor that maps every epoch to the empty set of actions. Note that the logic programming translation of the policy does not change. The only change occurs in the input where epochs are replaced with extended epochs. Furthermore, the (unchanged) logic program specifies now a maximal extended event-cancellation monitor. For an extended epoch E, let

$$occ'(E) = \{occ(e) | e \in E\}.$$

Theorem 5 For any coherent extended epoch E, E' is a maximally consistent reduction of E iff $E' = \{e | occ(e) \in M \wedge ignore(e) \notin M\}$ and M is a minimal model of $\Pi_P \cup \Psi_{AC} \cup A_P \cup B_P \cup occ'(E)$. Furthermore, E' is coherent since $E' \subseteq E$.

Define the monitor $\Omega'_{P,AC}$ like $\Omega_{P,AC}$ in Definition 12, with $occ'(E)$ substituted for $occ(E)$. From Theorem 5, it follows that $\Omega'_{P,AC}$ is a maximal extended event-cancellation monitor. In addition observe that if we extend a (non-extended) epoch E to an extended epoch $ext(E) = \{g | g \in E\} \cup \{!e | e \text{ has no instances in } E\}$, $\Omega_{P,AC}(E) = \Omega'_{P,AC}(ext(E))$. It should also be clear that the complexity bounds (Theorem 3) do not change.

Concerning Example 10, note that although the monitor that outputs a_1 for every epoch is a maximal event-cancellation monitor, it is not a maximal *extended* event-cancellation monitor. $\Omega'_{P_4,AC}$, which outputs the empty set for $\{e_1\}$ is both an *extended* event-cancellation and action-cancellation monitor. Similarly, in Example 9, there is now an extended event-cancellation monitor: one that outputs an empty set of actions for every epoch.

We conclude the discussion of negation by addressing an *expressiveness* issue. At first glance it appears that allowing negated events in the rules may obviate the need for action constraints. In Example 2, adding *!defectiveProduct* to the second and third rules makes the action constraint superfluous. However, the elimination of action constraints is not always possible. In Example 1, eliminating the constraint would require adding existential quantification to the event language.

6 Rule clusters

As we have seen earlier, in some cases the unobtrusiveness requirement for monitors is desirable, while in other it is not. Moreover, it appears useful to be able to enforce this requirement only for selected parts (rules) of a policy.

6.1 Clusters and partitions

We define a *rule cluster* of a $\mathcal{PDL}$ policy P to be a subset of the set of rules of P. A *cluster partition* of P is a set of disjoint rule clusters of P that cover all of P. For a cluster partition $\mathcal{P} = (P_1, \ldots, P_k)$ of P and a set of action constraints AC, a *partitioned monitor* produces for each cluster P_i an action set A_i that has to satisfy the following properties:

- $\bigcup_{1 \le i \le k} A_i$ satisfies AC,
- for every epoch E and every i, $1 \le i \le k$, there is an epoch $E_i \subseteq E$ such that $A_i = \pi_{P_i}(E_i)$,

The first condition guarantees that the partitioned monitor is indeed a monitor, i.e., satisfies action constraints. The second condition provides unobtrusiveness *within* a single rule cluster. Separate clusters are independent and an event can be ignored only in some clusters.

For a given cluster partition $\mathcal{P} = (P_1, \ldots, P_k)$ and a given epoch E, the partitioned monitors corresponding to $\mathcal{P}$ can be ordered using the component-wise order. Consider the partition $\mathcal{P}_0$ in which every cluster consists of a single rule of the policy P. A partitioned monitor corresponding to $\mathcal{P}_0$ which additionally is maximal in the above ordering is a maximal action-cancellation monitor of P. On the other hand, a maximal partitioned monitor corresponding to the partition $\mathcal{P}_1$ consisting of a single cluster is a maximal event-cancellation monitor of P. So by adjusting the partition of P we can achieve the desired degree of unobtrusiveness in the monitor.

6.2 Translation to logic programs

We show here how to obtain an executable specification of a maximal partitioned monitor of a policy by modifying the translation to logic programs defined earlier. We simulate the partitioning of rules into clusters by introducing an additional subscript, corresponding to the cluster number, into the monitor meta-predicates. However, only some predicates: *exec*, *ignore*, and *accept* need to be subscripted in that way. The remaining predicates: *occ* and *block* are interpreted identically in each cluster and thus do not need to be subscripted.

We show now how to obtain the translation of the rules of a policy P. Consider the policy translation

first. Each rule

$$e_1 \& \ldots \& e_l \textbf{ causes } a \textbf{ if } C$$

in a cluster P_i is translated into the implication

$$exec_i(a) \leftarrow occ(e_1) \wedge \ldots \wedge occ(e_l) \wedge C.$$

Additionally, for each cluster P_i there is a rule that collects the actions to be executed (A is a logical variable here):

$$exec(A) \leftarrow exec_i(A).$$

Denote the set of implications obtained in this way by Π_{P_i}. Second, the *conflict rules* Ψ_{AC} are defined as before. Thus, each constraint in AC of the form

$$\textbf{never } a_1 \wedge \ldots \wedge a_n \textbf{ if } C$$

is translated into the rule

$$block(a_1) \vee \ldots \vee block(a_n) \leftarrow exec(a_1) \wedge \ldots \wedge exec(a_n) \wedge C.$$

Third, for each rule

$$e_1 \& \ldots \& e_l \textbf{ causes } a \textbf{ if } C$$

in a cluster P_i, we obtain a *blocking rule*

$$ignore_i(e_1) \vee \ldots \vee ignore_i(e_n) \leftarrow occ(e_1) \wedge \ldots \wedge occ(e_n) \wedge C \wedge block(a)$$

and an *accepting rule*

$$accept_i(a) \leftarrow occ(e_1) \wedge \ldots \wedge occ(e_n) \wedge C \wedge \neg ignore_i(e_1) \wedge \ldots \wedge \neg ignore_i(e_n).$$

For a cluster P_i, denote the set of blocking rules by B_{P_i} and the set of accepting rules by A_{P_i}.

Theorem 6 Let $\mathcal{P} = (P_1, \ldots, P_k)$ be a cluster partition of a policy P. Let $\Omega_{\mathcal{P},AC} : Epochs \longrightarrow (2^{Actions})^k$ be a transducer defined for any epoch E in the following way:

1. Select a minimal model M of
$$\bigcup_i \Pi_{P_i} \cup \Psi_{AC} \cup \bigcup_i (A_{P_i} \cup B_{P_i} \cup \{r_{P_i}, c_{P_i}\}) \cup occ(E).$$
2. Return the vector $(A_1, \ldots, A_k)$ where $A_i = \{a | accept_i(a) \in M\}$ as output (i.e., $\Omega_{\mathcal{P},AC}(E) = \{a | accept_i(a) \in M\}$).

Then $\Omega_{\mathcal{P},AC}$ is a maximal partitioned monitor corresponding to $\mathcal{P}$.

If we have a policy that is to be monitored unobtrusively together with a rule for which this requirement is undesirable (as in Example 3), we just break the policy into two clusters and use the corresponding partitioned monitor.

Example 11 Consider again Example 2. If each rule is in a separate cluster and *mailProduct* is blocked, then the event *orderReceived* is ignored in the second cluster but not in the third one and the action *chargeCC* is accepted, negating unobtrusiveness.

For the purpose of efficient conflict resolution, a set of single rule clusters may be viewed as a single policy. For this policy a PTIME maximal action-cancellation monitor is obtained, via an adaptation of the proof of Theorem 2. The output of this monitor is a set of *accept* atoms and can be combined with the outputs of the remaining clusters.

7 The policy server

A network management system based on full $\mathcal{PDL}$ has been incorporated into a commercial softswitch developed at Bell Labs that couples public telephony with IP technology [22]. The softswitch manages an unbounded number of policy servers that are able to run policies written in $\mathcal{PDL}$.[1] When a policy is loaded into a policy server the server creates a policy evaluator for the input policy, it contacts the devices (i.e., routers, hubs, computers, etc.) that can potentially generate instances of the events of interest to the policy and registers the interest with the devices. The registration happens at policy enabling points (PEPs) that wrap around the devices to act as interfaces between the devices and the policy servers. Events generated by a device are intercepted by its assigned PEP, translated into event terms and sent to the appropriate policy server. When an event arrives at the policy server, the server gives copies of the event to each policy evaluator that is running a policy that mentions the event. Each evaluator accumulates the events in a buffer and using a time constant T given to the evaluator during initialization, the evaluator groups events from the buffer into epochs based on the following criterion. An event e arriving at a buffer at time T_1 belongs to the same epoch than the previous event in the buffer if the difference between the time of the beginning of the previous event epoch and T_1 is less than or equal to T. Otherwise, the arriving event belongs to a new epoch and the beginning time of this new epoch is set to T_1. For the special case in which e is

[1] It is bounded only by the capacity of the computers used.

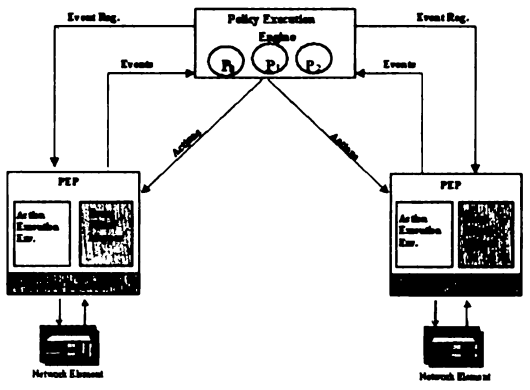


Figure 1: System Architecture

the first event sent to the evaluator, the first epoch is started with the beginning time also set to T_1. Each policy evaluator runs with the appropriate epoch as input. Following the policy evaluations, each policy evaluator returns a set actions. The server takes these actions and sends them to the appropriate PEPs which translate them into device-specific operations. The architecture of the server is depicted in Figure 1. The system is completely written in Java except for some parts of the PEPs which are device-dependent.

One possible way to add the conflict resolution capabilities described here to the current implementation is for the policy server to be extended as follows. For every set of actions A generated by the evaluation of a policy P with an input epoch E, take the epoch E and the set $exec(A) = \{exec(a)|a \in A\}$, and execute the following two steps:

1. Select a minimal model M of $exec(A) \cup \Psi_{AC} \cup A_P \cup B_P \cup occ(E)$.
2. Return the set $\{a|accept(a) \in M\}$ as output.

This new set will be used by the server as the final output of the evaluation of P. Observe that the difference between these two steps and the monitor $\Omega_{P,AC}$ introduced in 12 is that Π_P has been replaced by $exec(A)$. We are using the current implementation to evaluate the policy and leave only the resolution of conflicts to the logic program.

8 Related work

Several languages have been proposed for policy based network management (see for example [18, 24, 11, 14]). Much of the work in these papers is dedicated to implementation issues. Similarly, there are several implementations of event notification systems available but all have informal specifications (see for example [10] and the references therein). To address problems like conflict resolution it is essential to have languages with precise semantics like $\mathcal{PDL}$. However, the framework described in the present paper is not restricted to $\mathcal{PDL}$ and can be applied to other event languages. We will briefly discuss in the next section some general guidelines on how we can extend the framework to a more general class of policies.

More formal work on event notification languages has been developed specifically with the goal of network monitoring [25, 11]. The emphasis in these languages is on very expressive primitive events and event compositions. Actions are secondary, so the question of conflicts was never addressed. Our approach assumes that the user knows in advance the action constraints. However, if policies are developed by independent entities sometimes it might be difficult to obtain the constraints. Work on detecting statically potential conflicts is reported in [16]. This work might be useful to generate action constraints automatically.

Conflict resolution among production rules has been studied in AI and databases. For example, OPS5 [2] uses elaborate criteria depending on the form of the rules and the data to resolve such conflicts. Active database systems, on the other hand, typically use priorities to choose among conflicting rules [1]. A comprehensive framework for conflict resolution in this context was presented in [12]. The results about the computational complexity of testing consistency of production rules were presented in [3]. Those works are quite different from ours in that they assume interpreted actions (variable assignments or database updates) and mostly ignore the event part of event-condition-action rules. A recent work [13] deals with a model that is closer to ours, although the conflicts studied are still between the rules, not actions, and the events are not taken into account. This work proposes a metalanguage for the control of rule executions. The rules themselves are viewed as black boxes. Using the metalanguage of [13], one can often achieve similar effects to our framework. For example, the policy P_1 from Example 2 can be represented as a conjunction of the following relationships from [13]: the rules 1 and 2 are mutually exclusive, the rule 1 is preferred over the rule 2, the rules 2 and 3 require each other. The relative expressive power of the framework of [13] and our framework remains to be investigated. As should be clear from the above example, the framework of [13] is less declarative and lower-level than ours, and conflict resolution and rule dependencies have to be explicitly programmed-in.

The notion of *action constraints* was independently introduced in [5] and [8] (a similar notion was also proposed in [9]). In the paper [8], Eiter *et al* introduced a modal policy specification language for software agents. Conflict resolution is only one of the many issues addressed in [8]. The authors propose, using an entirely different terminology, maximal action cancellation monitors, without considering event cancellation. In general, conflict resolution methods in the current active rule literature, including [13] and [8], are not *unobtrusive* and in many cases the result they produce does not necessarily correspond to conflict-free executions of the rules. The paper [7] is a follow-up paper to [8] and contains numerous complexity results. They all involve, however, languages richer that the subset of $\mathcal{PDL}$ used in the present paper.

Several recent works [8, 16, 19] have postulated a modal, deontic framework for specifying prohibitions and obligations of agents in a distributed environment. Simple obligations and prohibitions can be captured in our framework. First, notice that an ECA rule can be read as an *obligation* to execute its action part if the events in the event part occur. Second, a *prohibition* to execute an action a if some events $e_1, \ldots, e_n$ occur can be simulated using action constraints and priorities (Section 3.3). We introduce the rule

$$e_1 \& \ldots \& e_n \textbf{ causes } not\_a$$

plus the constraint

$$\textbf{never } a \wedge not\_a$$

into the policy. The action not_a is a new action and has no effect. We also give priority to not_a over a.

9 Conclusions and further work

In this paper we have presented a logic programming framework for detecting action conflicts in policies and finding resolutions to these conflicts. The class of policies that we consider is limited to *stateless* transducers. We are currently extending our approach to deal with *state*. In particular, we accommodate *sequence events* (like in [20]), already a part of full $\mathcal{PDL}$ [15]. This extension requires generalizing the semantics of policies and monitors to mappings from *sequences* of epochs to *sequences* of sets of actions. Also, we are studying monitors that are not based on cancelling conflicting actions but rather on *delaying* them until no conflict occurs. Finally, we are considering a more general constraint language that permits *temporal* constraints. These extensions will enable us to deal with, among others, the issue of conflict resolution in active database rules. To address conflict resolution for full $\mathcal{PDL}$, we need to consider regular event expressions and aggregation. The generalized semantics, described above, is sufficient to handle this extension. However, the definition of the appropriate maximal monitors is non-trivial.

There is also another dimension of policy conflicts yet to be explored. Sometimes conflict between policies is resolved not by cancelling or delaying actions but by *composing* the policies into a new policy. Suppose, for example, clients make calls to a customer support line. Support agents are distributed around the world and a policy is written to route calls according to the geographic location of both the caller and the support agents, and the current congestion conditions of the network. There is also a policy that tries to match support agents with clients according to a profile of the client needs. A monitor, perhaps with some extra information, should make a decision on how to connect clients to agents taking into account both policies. Policies of this type are typical of telephone call centers. They are hard-coded in the center and conflicts are currently solved manually by modifying the policies involved.

References

[1] R. Agrawal, R. Cochrane, and B. G. Lindsay. On maintaining priorities in a production rule system. In *VLDB*, pages 479–487, 1991.

[2] L. Brownston, R. Farell, E. Kant, and N. Martin. *Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming.* Addison-Wesley, 1985.

[3] H. Kleine Büning, U. Löwen, and S. Schmitgen. Inconsistency of production systems. *Journal of Data and Knowledge Engineering*, 3:245–260, 1988/89.

[4] S. Ceri and P. Fraternali. *Designing Database Applications with Objects and Rules: The IDEA Methodology.* Addison-Wesley, 1997.

[5] J. Chomicki, J. Lobo, and S. Naqvi. Axiomatic conflict resolution in policy management. Technical Report ITD-99-36448R, Bell Labs, February 1999.

[6] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive datalog. *ACM Transactions on Database Systems*, 22(3), 1997.

[7] T. Eiter and V.S. Subrahmanian. Heterogeneous active agents, II: Algorithms and complexity. *Artificial Intelligence*, 108:257–307, March 1999.

[8] T. Eiter, V.S. Subrahmanian, and G. Pick. Heterogeneous active agents, I: Semantics. *Artificial Intelligence*, 108:179–255, March 1999.

[9] B. N. Grosof, Y. Labrou, and H. Y. Chan. A declarative approach to business rules in contracts: Courteous logic programs in XML. In Michael P. Wellman, editor, *Proc. of the First ACM Conference on Electronic Commerce*. ACM Press, November 1999.

[10] R. E. Gruber, B. Krishnamurthy, and E. Panagos. High-level constructs in the READY event notification system. In *8th ACM SIGOPS European Workshop*, Sintra, Portugal, September 1998.

[11] M. Z. Hasan. An active temporal model for network management databases. In *IFIP/IEEE 4th International Symposium on Integrated Network Management*, pages 524–535, Santa Barbara, California, May 1995.

[12] Y. E. Ioannidis and T. K. Sellis. Supporting inconsistent rules in database systems. *Journal of Intelligent Information Systems*, 1(3/4), 1992.

[13] H. V. Jagadish, A. O. Mendelzon, and I. S. Mumick. Managing conflicts between rules. In *Proc. 15th ACM SIGACT/SIGMOD Symposium on Principles of Database Systems*, pages 192–201, 1996.

[14] T. Koch, C. Krell, and B. Krämer. Policy definition language for automated management of distributed systems. In *Second International Workshop on Systems Management*, Toronto, Canada, June 1996.

[15] J. Lobo, R. Bhatia, and S. Naqvi. A policy description language. In *Proc. of AAAI*, Orlando, FL, July 1999.

[16] E. C. Lupu and M. Sloman. Conflict analysis for management policies. In R. Stadler A. Lazar, R. Saraco, editor, *Proc. 5th IFIP/IEEE International Symposium on Integrated Network Management*, pages 430–443, 1997.

[17] N. H. Minsky and V. Ungureanu. A mechanism for establishing policies for electronic commerce. In *Proc. of the 18th International Conference on Distributed Computing Systems*, May 1998.

[18] J. Moffett and M. S. Sloman. Policy hierarchies for distributed system management. *IEEE JSAC*, 11(9), 1993.

[19] J. D. Moffett and M. S. Sloman. Policy conflict analysis in distributed system management. *Journal of Organizational Computing*, 4(1):11–22, 1994.

[20] I. Motakis and C. Zaniolo. Temporal aggregation in active database rules. In *Proc. of SIGMOD*, Tucson, AZ, May 1997.

[21] I. Niemelä. Towards efficient default reasoning. In *Proc. of 14th International Joint Conference on Artificial Intelligence*, pages 312–318, Montreal, 1995.

[22] A. Virmani, J. Lobo, and M. Kohli. NETMON: Network management for the SARAS softsiwtch. In *Proc. of the IEEE/IFIP Network Operations and Management Symposium*, April 2000. To appear.

[23] J. Widom and S. Ceri. *Active Database Systems*. Morgan-Kaufmann, 1995.

[24] R. Wies. Policies in network and system management - formal definition and architecture. *Journal of Network and System Management*, 2(1):63–83, 1994.

[25] O. Wolfson, S. Sengupta, and Y. Yemini. Managing communication networks by monitoring databases. *IEEE Transactions on Software Engineering*, 17(9):944–953, September 1991.

Integration of Knowledge Sources

On the Difference between Merging Knowledge Bases and Combining them

Sébastien Konieczny
Laboratoire d'Informatique Fondamentale de Lille
Université de Lille 1 - 59655 Villeneuve d'Ascq - France
konieczn@lifl.fr

Abstract

We investigate the logical properties of knowledge base combination operators proposed in the literature. These operators are based on the selection of some maximal subsets of the union of the knowledge bases. We argue that they are not fully satisfactory to merge knowledge bases, since the source of information is lost in the combination process. We show that it is the reason why those operators do not satisfy a lot of logical properties. Then we propose to use more refined selection mechanisms in order to take the distribution of information into account in the combination process. That allows to define merging operators with a more subtle behaviour.

1 INTRODUCTION

In the fields of artificial intelligence and databases, one is often faced with conflicting information coming from several sources. Thus, an important problem in such cases is how to reach a coherent piece of information from these contradictory ones.

For example, if one wants to build an expert system from a group of human experts, it is sensible to code the knowledge of each expert in a knowledge base and then to combine them in a knowledge base that represents the knowledge of the group. This process allows to discover new pieces of knowledge distributed among the sources. For example, if an expert knows that a is true and another one knows that $a \rightarrow b$ holds, then the "synthesized" knowledge knows that b is true whereas none of the expert knows it. This is called *implicit knowledge* in [HM92]. However, simply put these knowledge bases together is a wrong way since there could be contradictions between some experts.

Some combination operators have been proposed, see *e.g.* [BKM91, BKMS92]. They are all based on the union of all the knowledge bases and on the selection of some maximal subsets, due to a given order (not necessarily the inclusion).

We study the logical properties of these operators. More exactly we investigate the rationality of these operators through the logical characterization of merging operators stated in [KP98, KP99]. This characterization is useful to classify particular merging methods and to highlight flaws and advantages of each of them.

In particular, an important drawback of combination operators is that the source of each knowledge is lost in the fusion process. We shall call merging operators the fusion operators that take the source of information into account. We propose in this paper a definition of selection functions *à la* AGM [AGM85, Gär88] for combination operators that allows to take into account the source of each piece of information. So we can define operators with a more subtle behaviour.

In order to motivate the need to take the source of information into account, consider the following scenario: Consider that we want to combine the following knowledge bases: $K_1 = K_2 = \{a, b\}$, $K_3 = \{a, b \rightarrow c\}$, $K_4 = \{\neg a, d\}$. Then the union of the knowledge bases is $\{a, \neg a, b, b \rightarrow c, d\}$. With a combination operator the maxiconsistent sets will be $\{a, b, b \rightarrow c, d\}$ and $\{\neg a, b, b \rightarrow c, d\}$.With this result we can not decide whether a or $\neg a$ holds. But a is supported by three of the four experts whereas only one supports $\neg a$. So it could be sensible to put a in the resulting knowledge base. Combination operators do not allow to take such arguments into account. We will see how to build merging operators that allow that behaviour.

The rest of the paper is organized as follows: In section 2 we give some definitions and state some notations. In section 3 we give a set of logical properties for merging operators. In section 4 we investigate the logical proper-

ties of some combination operators given in [BKMS92]. In section 5 we propose to use selection functions in order to define merging operators taking care of the distribution of information among the sources. Finally we discuss open problems in a concluding section.

2 PRELIMINARIES

Definition 1 *A knowledge base K is a finite set of well-formed first-order formulae.*

Note that a knowledge base is not necessarily closed under consequence relation.

Definition 2 *A knowledge set E is a multi-set of knowledge bases.*

We will note $\sqcup$ the union on multi-sets. By abuse if K is a knowledge base, K will also denote the knowledge set $E = \{K\}$. And if K and K' are two knowledge bases we will denote $K \sqcup K'$ the knowledge set $E = \{K, K'\}$.

For the proofs, we need to define a partition operator $\oplus$. $K = K' \oplus K''$ denotes that $K' \cup K'' = K$ and that $K' \cap K'' = \emptyset$.

Let K and K' be two knowledge bases, $K \wedge K'$ will denote the knowledge base $K \cup K'$. Analogously, if $E = \{K_1, \ldots, K_n\}$, then $E \wedge K = \{K_1 \wedge K, \ldots, K_n \wedge K\}$. We will note $\bigwedge E$ the conjunction of the knowledge bases of E, *i.e.* $\bigwedge E = K_1 \wedge \ldots \wedge K_n$.

Definition 3 *A knowledge set E is consistent if and only if $\bigwedge E$ is consistent.*

In addition to these basic definitions, we have to define the equivalence between knowledge sets.

Definition 4 *Let E_1, E_2 be two knowledge sets. E_1 and E_2 are equivalent, noted $E_1 \leftrightarrow E_2$, iff there exists a bijection f from $E_1 = \{K_1^1, \ldots, K_n^1\}$ to $E_2 = \{K_1^2, \ldots, K_n^2\}$ such that $f(K) \leftrightarrow K$.*

Let E be a knowledge set, E^n will denote the knowledge set $\underbrace{E \sqcup \ldots \sqcup E}_{n}$.

The result of the combination operators investigated in this paper is a set of knowledge bases. These sets have been called *flocks* by Fagin *et al.* [FKUV86].

Note that flocks and knowledge sets are both sets of knowledge bases (in fact knowledge sets are multi-sets). But the difference is that in the case of knowledge sets, the sets denote different sources of information, whereas in flocks the sets denote alternatives about the result of a combination. Flocks are similar to the extensions in default knowledge bases [Rei80, Eth88]. In order to underline this difference we will note K the elements of a knowledge set and M, P, Q, R the elements of a flock.

In order to investigate the logical properties of combination operators we need to define what are the consequences of a flock. We will adopt a cautious approach that considers, in a sense, flocks as disjunctions of knowledge bases.

Definition 5 *Let $\mathcal{F} = \langle M_1, \ldots, M_n \rangle$ be a flock. If $\mathcal{F} = \emptyset$, then $\mathcal{F}$ is inconsistent and, as usual, $Cn(\mathcal{F})$ is the set of all formulae. Else we define*

$$Cn(\mathcal{F}) = \bigcap_{i=1}^{n} Cn(M_i)$$

We can then define equivalence between flocks:

Definition 6 *Let $\mathcal{F}$ and $\mathcal{F}'$ be two flocks, we say that $\mathcal{F}$ implies $\mathcal{F}'$, noted $\mathcal{F} \vdash \mathcal{F}'$ if $Cn(\mathcal{F}) \supseteq Cn(\mathcal{F}')$. $\mathcal{F}$ and $\mathcal{F}'$ are equivalent, noted $\mathcal{F} \equiv \mathcal{F}'$, if both $\mathcal{F} \vdash \mathcal{F}'$ and $\mathcal{F}' \vdash \mathcal{F}$ hold. Similarly, we define $\mathcal{F} \vdash K$ where K is a knowledge base as $Cn(\mathcal{F}) \supseteq Cn(K)$.*

Definition 7 *Let $\mathcal{F} = \langle M_1, \ldots, M_n \rangle$ and $\mathcal{F}' = \langle M'_1, \ldots, M'_m \rangle$ be two flocks. $\mathcal{F} \vee \mathcal{F}'$ denotes the flock $\langle M_1, \ldots, M_n, M'_1, \ldots, M'_m \rangle$ and $\mathcal{F} \wedge \mathcal{F}'$ denotes the flock $\langle M''_{1,1}, \ldots, M''_{n,m} \rangle$ where $M''_{j,k} = M_j \cup M'_k$.*

So notice that if $\mathcal{F} = \langle M_1, \ldots, M_n \rangle$ and $\mathcal{F}' = \mathcal{F} \vee \langle P_1, \ldots, P_m \rangle$ with $\forall i$ P_i inconsistent, then $\mathcal{F} \equiv \mathcal{F}'$. So, when considering a flock, we can focus only on its consistent knowledge bases.

3 IC MERGING OPERATORS

In [KP98, KP99] a set of logical properties for merging operators is stated. We call operators satisfying these postulates Integrity Constraints merging operators (IC merging operators for short). Next we recall those postulates.

Definition 8 *Let E be a knowledge set, let IC be a knowledge base coding the integrity constraints of the merging, and let Δ be an operator that assigns to each knowledge set E and knowledge base IC a knowledge base $\Delta_{IC}(E)$. Δ is an IC merging operator if and only if it satisfies the following properties:*

(IC0) $\Delta_{IC}(E) \vdash IC$

(IC1) *If IC is consistent, then $\Delta_{IC}(E)$ is consistent*

(IC2) *If $\bigwedge E$ is consistent with IC, then* $\Delta_{IC}(E) = \bigwedge E \wedge IC$

(IC3) *If* $E_1 \leftrightarrow E_2$ *and* $IC_1 \leftrightarrow IC_2$, *then* $\Delta_{IC_1}(E_1) \leftrightarrow \Delta_{IC_2}(E_2)$

(IC4) *If* $K \vdash IC$ *and* $K' \vdash IC$, *then* $\Delta_{IC}(K \sqcup K') \wedge K \nvdash \bot \Rightarrow \Delta_{IC}(K \sqcup K') \wedge K' \nvdash \bot$

(IC5) $\Delta_{IC}(E_1) \wedge \Delta_{IC}(E_2) \vdash \Delta_{IC}(E_1 \sqcup E_2)$

(IC6) *If* $\Delta_{IC}(E_1) \wedge \Delta_{IC}(E_2)$ *is consistent, then* $\Delta_{IC}(E_1 \sqcup E_2) \vdash \Delta_{IC}(E_1) \wedge \Delta_{IC}(E_2)$

(IC7) $\Delta_{IC_1}(E) \wedge IC_2 \vdash \Delta_{IC_1 \wedge IC_2}(E)$

(IC8) *If* $\Delta_{IC_1}(E) \wedge IC_2$ *is consistent, then* $\Delta_{IC_1 \wedge IC_2}(E) \vdash \Delta_{IC_1}(E)$

Most of these postulates are a generalization of belief revision postulates [AGM85, Gär88, KM91]. (IC0) states that the result of the merging complies with the integrity constraints. (IC1) ensures that, when the integrity constraints are consistent, we always manage to extract a coherent piece of information from the knowledge set. (IC2) says that, if it is possible, the result of the merging is simply the conjunction of the knowledge bases of the knowledge set with the integrity constraints. (IC3) is the principle of irrelevance of syntax. It states that if two knowledge sets are equivalent and two integrity constraints knowledge bases are equivalent, then the result of the merging of each knowledge set under their respective integrity constraints will give two equivalent knowledge bases. The purely "merging" postulates are (IC4),(IC5) and (IC6). (IC4) is what we call the fairness postulate. It ensures that when one merges two knowledge bases, it can not give the preference to one of them. (IC5) and (IC6) correspond to Pareto's conditions in Social Choice Theory [Arr63]. (IC5) states that if a group compromises on a set of alternatives A belongs to, and another group compromises on another set of alternatives which contains also A, then A has to be in the chosen alternatives if we join the two groups. (IC6) states that if a group prefers strictly an alternative A to an alternative B and another group finds A and B equally plausible, then A will be preferred to B if we join the two groups. Finally (IC7) and (IC8) state conditions on the conjunction of integrity constraints. It ensures that the notion of "closeness" is well-behaved. See [KP99] for a full motivation of this set of postulates and for a semantical characterization in terms of family of pre-orders on interpretations.

There are two major subclasses of merging operators, namely majority and arbitration operators. Whereas majority operators try to satisfy the majority of the protagonists, arbitration operators try to satisfy each protagonist to the best possible degree.

A majority merging operator is an IC merging operator that satisfies the following property:

(Maj) $\exists n \ \Delta_{IC}(E_1 \sqcup E_2^n) \vdash \Delta_{IC}(E_2)$

This postulate expresses the fact that if an opinion has a large audience, it will be the opinion of the group.

An arbitration operator is an IC merging operator that satisfies the following property:

$$\textbf{(Arb)} \quad \left.\begin{array}{l} \Delta_{IC_1}(K_1) \leftrightarrow \Delta_{IC_2}(K_2) \\ \Delta_{IC_1 \leftrightarrow \neg IC_2}(K_1 \sqcup K_2) \leftrightarrow (IC_1 \leftrightarrow \neg IC_2) \\ IC_1 \nvdash IC_2 \\ IC_2 \nvdash IC_1 \end{array}\right\} \Rightarrow \quad \Delta_{IC_1 \vee IC_2}(K_1 \sqcup K_2) \leftrightarrow \Delta_{IC_1}(K_1)$$

From a semantical point of view (Arb) ensures that it is the median possible worlds that are chosen, that is if K_1 prefers strictly a world A to a world B and if K_2 prefers strictly A to a world C and if B and C are equally desirable for the merging, then A will be strictly preferred to B and C for the merging (*cf* [KP99]).

Another property, opposed to the majority postulate, we can mention is the *majority independence* which is the following one:

(MI) $\forall n \ \Delta_{IC}(E_1 \sqcup E_2^n) \leftrightarrow \Delta_{IC}(E_1 \sqcup E_2)$

That very strong property states that the result of the merging is fully independent of the popularity of the views but simply takes into account each different view. But the following results hold [KP98]:

Theorem 1 *(i). There is no IC merging operator satisfying (MI).*
(ii). If an operator satisfies (IC1), then it can't satisfy both of (MI) and (Maj).

4 COMBINATION OPERATORS

Baral, Kraus, Minker and Subrahmanian proposed in [BKM91, BKMS92] several theory merging operators, these operators are based on a selection of maxiconsistent subsets in the union of the knowledge bases of the knowledge set.

Once the union of the knowledge bases is settled, the problem is to find a coherent information from an inconsistent knowledge base. Thus, such a definition is very close to Brewka's preferred subtheories [Bre89] and to the work of Benferhat *et al.* on entailment in inconsistent databases [BCD+93, BDP97, BDL+98].

Definition 9 *Let* MAXCONS(K, IC) *be the set of maximal (with respect to inclusion) consistent subsets of*

$K \wedge IC$ which contain IC, i.e. MAXCONS(K, IC) *is the set of all M such that*

- $M \subseteq K \wedge IC$,
- $IC \subseteq M$,
- *if* $M \subset M' \subseteq K \wedge IC$, *then* $M' \vdash \bot$.

Let MAXCONS(E, IC) = MAXCONS$(\bigwedge E, IC)$. *We will use the subscript* MAXCONS$_{card}(E, IC)$ *when the maximality of the sets is with respect to cardinality.*

Let's define the following operators:

Definition 10 *Let E be a knowledge set and IC be a knowledge base:*

$\Delta^{C1}{}_{IC}(E)$ = MAXCONS(E, IC)
$\Delta^{C3}{}_{IC}(E) = \{M : M \in$ MAXCONS$(E, \top)$ *and* $M \wedge IC$ *consistent*$\}$
$\Delta^{C4}{}_{IC}(E) = \{M : M \in$ MAXCONS$_{card}(E, IC)\}$
$\Delta^{C5}{}_{IC}(E) = \{M \wedge IC : M \in$ MAXCONS$(E, \top)$ *and* $M \wedge IC$ *consistent* $\}$ *if this set is non empty and IC otherwise.*

The $\Delta^{C1}{}_{IC}(E)$ operator takes as result of the combination the set of maximal consistent subsets of $E \wedge IC$ which contain the constraints IC. The $\Delta^{C3}{}_{IC}(E)$ operator computes first the set of maximal consistent subsets of E, and then selects those that are consistent with the constraints. The $\Delta^{C4}{}_{IC}(E)$ operator selects the set of consistent subsets of $E \wedge IC$ which contain the constraints IC and that are maximal with respect to cardinality.

The operators $\Delta^{C1}{}_{IC}(E)$, $\Delta^{C3}{}_{IC}(E)$ and $\Delta^{C4}{}_{IC}(E)$ correspond respectively to the operators $Comb1(E, IC)$, $Comb3(E, IC)$ and $Comb4(E, IC)$ in [BKMS92]. The Δ^{C5} operator is a slight modification of Δ^{C3} in order to grasp more logical properties.

In the following theorems we investigate the logical properties of the operators defined above.

Theorem 2 *The Δ^{C1} operator satisfies (IC0), (IC1), (IC2), (IC4), (IC5), (IC7), and (MI). It does not satisfy (IC3), (IC6), (IC8) and (Maj).*

Proof: (IC0), (IC1) and (IC2) are satisfied by definition of the Δ^{C1} operator.

Δ^{C1} satisfies (IC4) because the stronger following property is satisfied:

If $K \vdash IC$ then $\Delta^{C1}{}_{IC}(K \sqcup K') \wedge K \nvdash \bot$.

Since K is a consistent subset of $K \wedge K' \wedge IC$ and by hyp. $K \vdash IC$, then there exists a maxiconsistent subset of $K \wedge K' \wedge IC$ that contains K. And then $\Delta^{C1}{}_{IC}(K \sqcup K') \wedge K \nvdash \bot$.

(IC5) holds. It is trivially true when $\Delta_{IC}(E_1) \wedge \Delta_{IC}(E_2)$ is not consistent. And if $\Delta_{IC}(E_1) \wedge \Delta_{IC}(E_2)$ is consistent, let P_i be the elements of MAXCONS$(E_1 \sqcup E_2, IC)$, Q_i the elements of MAXCONS(E_1, IC), and R_i the elements of MAXCONS(E_2, IC). To show that (IC5) holds it is enough to prove that if $Q_j \wedge R_k$ is consistent then $\exists i\ P_i = Q_j \wedge R_k$. First put $L = IC \cup \bigwedge E_1 \cup \bigwedge E_2$, $Q_j \wedge R_k$ is a consistent (with IC) subset of L, and as P_i is a maxiconsistent (with IC) subset of L, then $\exists i\ P_i \supseteq Q_j \wedge R_k$. We claim that $P_i \subseteq Q_j \wedge R_k$ holds. Because otherwise we have $P_i \supset Q_j \wedge R_k$. Then we can decompose $P_i = P_1 \wedge P_2$ with $P_1 = P_i \cap (\bigwedge E_1 \wedge IC)$ and $P_2 = P_i \cap (\bigwedge E_2 \wedge IC)$. And then we have that $P_1 \supset Q_j$ or $P_2 \supset R_k$. So either Q_j or R_k is not a maximum consistent subset of E_1 or E_2 respectively. Contradiction.

Δ^{C1} does not satisfy (IC6). Consider the following example: $K_1 = \{a \rightarrow c,\ e \rightarrow \neg c,\ b \rightarrow \neg c\}$, $K_2 = \{a, e\}$, $K_3 = \{b\}$ and $IC = \top$. Then $\Delta^{C1}(K_1 \sqcup K_2) \wedge \Delta^{C1}(K_3)$ is consistent but $\Delta^{C1}(K_1 \sqcup K_2 \sqcup K_3) \nvdash \Delta^{C1}(K_1 \sqcup K_2) \wedge \Delta^{C1}(K_3)$.

(IC7) is satisfied. When $\Delta_{IC_1}(E) \wedge IC_2$ is not consistent (IC7) is trivial. Otherwise let P_i be the elements of MAXCONS(E, IC_1) and Q_i the elements of MAXCONS$(E, IC_1 \wedge IC_2)$. It is enough to prove that if $P_i \wedge IC_2$ is consistent then $\exists Q_j$ such that $P_i \wedge IC_2 = Q_j$. Let $P_i \wedge IC_2$ be a consistent subset of $\bigwedge E \wedge IC_1 \wedge IC_2$ then there exists a maxiconsistent subset Q_j that contains $P_i \wedge IC_2$, so $\exists j\ P_i \subseteq Q_j$. Moreover, we have that $P_i \wedge IC_2 \supseteq Q_j$, otherwise $P_i \wedge IC_2 \subset Q_j$, and from this it is easy to see that $P_i \subset Q_j \cap (\bigwedge E \wedge IC_1)$. So P_i is not maximum. Contradiction.

(IC8) is not satisfied. We use the counterexample to (IC6) slightly modified: $K = \{a \rightarrow c, e \rightarrow \neg c, b \rightarrow \neg c\}$, $IC_1 = \{a, e\}$ and $IC_2 = \{b\}$. Then $\Delta^{C1}{}_{IC_1}(K) \wedge IC_2$ is consistent but $\Delta^{C1}{}_{IC_1 \wedge IC_2}(K) \nvdash \Delta^{C1}{}_{IC_1}(K) \wedge IC_2$.

■

Theorem 3 *The Δ^{C3} operator satisfies (IC4), (IC5), (IC7), (IC8), (MI). It does not satisfy (IC0), (IC1), (IC2), (IC3), (IC6) and (Maj).*

Proof: Δ^{C3} does not satisfy (IC0) since the result is only consistent with IC. It does not satisfy (IC1) since when there is no maxiconsistent consistent with IC the disjunction is empty and the result is $\bot$. And it does not satisfy (IC2), we have instead: If $\bigwedge E$ is consistent with IC, then $\Delta^{C3}{}_{IC}(E) = \bigwedge E$.

The proofs of (IC4) and (IC5) for Δ^{C3} are similar to the ones for Δ^{C1}.

Table 1: Properties of combination operators

	IC0	IC1	IC2	IC3	IC4	IC5	IC6	IC7	IC8	MI	Maj
Δ^{C1}	✓	✓	✓	–	✓	✓	–	✓	–	✓	–
Δ^{C3}	–	–	–	–	✓	✓	–	✓	✓	✓	–
Δ^{C4}	✓	✓	✓	–	–	–	–	✓	✓	✓	–
Δ^{C5}	✓	✓	✓	–	✓	✓	–	✓	✓	✓	–

Δ^{C3} does not satisfy (IC6). There is the same counterexample than for Δ^{C1}.

(IC7) and (IC8) are satisfied directly. If ${\Delta^{C3}}_{IC_1}(E) \wedge IC_2$ is not consistent (IC7) is trivial. Otherwise since by definition ${\Delta^{C3}}_{IC_1}(E) = \{Q : Q \in \text{MAXCONS}(E, \top)$ and $Q \wedge IC_1$ consistent$\}$, then ${\Delta^{C3}}_{IC_1}(E) \wedge IC_2 \equiv \{Q : Q \in \text{MAXCONS}(E, \top)$ and $Q \wedge IC_1 \wedge IC_2$ consistent$\}$ (this set is non empty by hypothesis) what is by definition ${\Delta^{C3}}_{IC_1 \wedge IC_2}(E)$. ■

Theorem 4 *The Δ^{C4} operator satisfies (IC0), (IC1), (IC2), (IC7), (IC8) and (MI). It does not satisfy (IC3), (IC4), (IC5), (IC6) and (Maj).*

Proof: (IC0), (IC1) and (IC2) are satisfied by definition.

Δ^{C4} does not satisfy (IC4): Let's take $IC = \{a, b\}$, $K = \{a, b, c, d\}$, $K' = \{a, b, c \rightarrow \neg d, d \rightarrow \neg c\}$. Then ${\Delta^{C4}}_{IC}(K \sqcup K') = \{\{a, b, c, c \rightarrow \neg d, d \rightarrow \neg c\}, \{a, b, d, c \rightarrow \neg d, d \rightarrow \neg c\}\}$, so ${\Delta^{C4}}_{IC}(K \sqcup K') \wedge K' \not\vdash \bot$ and ${\Delta^{C4}}_{IC}(K \sqcup K') \wedge K \vdash \bot$.

(IC5) does not hold. Consider the following example: Let $K_1 = \{a\}$, $K_2 = \{\neg a \wedge b\}$, $K_3 = \{\neg a \wedge c\}$. Then with $E_1 = K_1 \sqcup K_2$, $E_2 = K_1 \sqcup K_3$ and $IC = \top$, we have ${\Delta^{C4}}_{IC}(E_1) \wedge {\Delta^{C4}}_{IC}(E_2) \not\vdash {\Delta^{C4}}_{IC}(E_1 \sqcup E_2)$.

(IC6) is not satisfied. Consider the following example: Let $K_1 = \{a\}$, $K_2 = \{\neg a \wedge (b \vee c)\}$, $K_3 = \{a, a \wedge z\}$, $K_4 = \{\neg a \wedge \neg b, \neg a \wedge \neg c\}$. Then with $E_1 = K_1 \sqcup K_2$, $E_2 = K_3 \sqcup K_4$ and $IC = \top$, we have ${\Delta^{C4}}_{IC}(E_1) \wedge {\Delta^{C4}}_{IC}(E_2)$ consistent but ${\Delta^{C4}}_{IC}(E_1 \sqcup E_2) \not\vdash {\Delta^{C4}}_{IC}(E_1) \wedge {\Delta^{C4}}_{IC}(E_2)$.

(IC7) and (IC8) are satisfied. When ${\Delta^{C4}}_{IC_1}(E) \wedge IC_2$ is not consistent (IC7) and (IC8) are satisfy straightforwardly. So assume that ${\Delta^{C4}}_{IC_1}(E) \wedge IC_2$ is consistent. Let P be an element of $\text{MAXCONS}_{card}(E, IC_1 \wedge IC_2)$. And let Q be an element of $\text{MAXCONS}_{card}(E, IC_1)$ consistent with IC_2. We want to show that $Q \cup IC_2 \in \text{MAXCONS}_{card}(E, IC_1 \wedge IC_2)$ and that P can be rewritten $P_1 \oplus P_2$ with $P_1 \in \text{MAXCONS}_{card}(E, IC_1)$ and $P_2 \subseteq IC_2$. This is enough to show that ${\Delta^{C4}}_{IC_1}(E) \wedge IC_2 \leftrightarrow {\Delta^{C4}}_{IC_1 \wedge IC_2}(E)$. Let's define $A_1 = \bigwedge E \cup IC_1$ and $A_2 = IC_2 \setminus A_1$. Then we can split $P = P_1 \oplus P_2$ with $P_1 = P \cap A_1$ and $P_2 = P \cap A_2$. Similarly let's define $Q \cup IC_2 = Q_1 \oplus Q_2$ such that $Q_1 = (Q \cup IC_2) \cap A_1$ and $Q_2 = (Q \cup IC_2) \cap A_2$. As $IC_2 \subseteq P$ and $IC_2 \subseteq Q \cup IC_2$, by construction it is easy to see that $P_2 = A_2 = Q_2$. In terms of cardinalities $|P| = |P_1| + |P_2|$ and $|Q \cup IC_2| = |Q_1| + |Q_2|$. But we have that $|P_2| = |Q_2|$. As P is in $\text{MAXCONS}_{card}(E, IC_1 \wedge IC_2)$ and $Q \cup IC_2 \subseteq E \cup IC_1 \cup IC_2$, then $|P| \geq |Q \cup IC_2|$. So $|P_1| \geq |Q_1|$. Similarly as $Q_1 = Q$ is in $\text{MAXCONS}_{card}(E, IC_1)$ and $P_1 \subseteq E \cup IC_1$, then $|P_1| \leq |Q_1|$. From this it is easy to see that both $|P| = |Q \cup IC_2|$ and $|Q| = |P_1|$ hold. So $Q \cup IC_2$ is in $\text{MAXCONS}_{card}(E, IC_1 \wedge IC_2)$, and P_1 is in $\text{MAXCONS}_{card}(E, IC_1)$. ■

Theorem 5 *The Δ^{C5} operator satisfies (IC0), (IC1), (IC2), (IC4), (IC5), (IC7), (IC8), (MI). It does not satisfy (IC3), (IC6) and (Maj).*

Proof: The proofs are mainly the same that for theorem 3 except (IC0), (IC1) and (IC2) that are now satisfied by definition. ■

We sum up the previous results in Table 1. The symbol ✓ (respectively –) in a square means that the corresponding operator satisfies (resp. does not satisfy) the corresponding postulate. By construction all the operators satisfy (MI). We can also note that none of these operators satisfies (IC6). We will see in the next section how to build merging operators with more logical properties.

None of the operators we study in this paper satisfies (IC3) since they are all syntax sensitive. We can illustrate this on the following example. Consider three knowledge bases $K_1 = \{a, b\}$, $K_2 = \{a \wedge b\}$, and $K_3 = \{\neg b\}$. Let $E_1 = K_1 \sqcup K_3$ and $E_2 = K_2 \sqcup K_3$ be two knowledge sets. The maxiconsistent subsets of E_1 are $\{a, b\}$ and $\{a, \neg b\}$, the ones of E_2 are $\{a \wedge b\}$ and $\{\neg b\}$. So each maxiconsistent of E_1 implies a, whereas it is not the case for E_2. So, although the knowledge bases K_1 and K_2 are logically equivalent, with the syntactical operators studied in this paper, the result of the fusion of E_1 will imply a, whereas it will not be the case with the fusion of E_2.

5 SELECTION FUNCTION AND MERGING OPERATORS

The combination operators do not take into account the "individual side" of the merging, since the source of information doesn't matter in the combination process. They simply put all the pieces of information together and then select some maximal consistent subsets. So, with this approach, it is not possible to try to reach the best consensus between protagonists. We can not for example select only the maxiconsistent sets that fit the majority of agents. In the same way we can not try to arbitrate these views, that is to satisfy all the protagonists to the best possible degree. The idea in this section is to use a selection function to choose among the maxiconsistent subsets, those that best fit a "merging criterion".

The motivation to define such selection functions comes from AGM revision framework [AGM85, Gär88]. The Δ^{C1} operator corresponds to full meet contraction function [AM82, Gär88] that has been shown to be unsatisfactory for a revision since it drops too much information. Partial meet contraction functions, defined from selection functions by choosing only some of the maxiconsistents, have been shown to have a less drastic behaviour.

In this section, we will examine some selection merging operators. The idea of this kind of operators is to give the preference to the maxiconsistents that are closer to the agents' view. The differences between these operators lie firstly in the definition of the "distance" between a maxiconsistent and a knowledge base, and secondly in the aggregation of these results to define the "distance" between a maxiconsistent and a knowledge set.

We will focus on operators defined from the Δ^{C1} operator and investigate their logical properties. But the following methods can also be used with the other combination operators.

5.1 DRASTIC MAJORITY OPERATOR

The "distance" between knowledge bases we will consider here is drastic. We will set this distance to 0 if the conjunction of these two knowledge bases is consistent and 1 otherwise.

Definition 11 *Consider a knowledge set E and a knowledge base M.*

- $dist_{\mathrm{D}}(M, K) = \begin{cases} 0 & \text{if } M \wedge K \text{ consistent} \\ 1 & \text{otherwise} \end{cases}$
- $dist_{\mathrm{D}}(M, E) = \sum_{K \in E} dist_{\mathrm{D}}(M, K)$
- $\Delta^{\mathrm{D}}_{IC}(E) = \{M \in \Delta^{C1}{}_{IC}(E) : dist_{\mathrm{D}}(M, E) = \min_{M_i \in \Delta^{C1}{}_{IC}(E)} (dist_{\mathrm{D}}(M_i, E))\}$

So this selection function chooses among the maxiconsistents those that are consistent with a maximum of knowledge bases.

It is easy to see that when a maxiconsistent is consistent with a knowledge base it contains this knowledge base. So it amounts to choose the maxiconsistents that contain a maximum of knowledge bases.

Theorem 6 *The Δ^{D}_{IC} operator satisfies (IC0), (IC1), (IC2), (IC4), (IC5), (IC7) and (Maj). It does not satisfy (IC3), (IC6), (IC8) and (MI).*

Proof: (IC0), (IC1) and (IC2) are straightforwardly satisfied.

Δ^{D} satisfies (IC4). Since either K is consistent with K' and then (IC4) holds trivially, or K is not consistent with K'. So there is no maxiconsistent consistent with the two knowledge bases. There exists a maxiconsistent that contains K. So $\Delta^{\mathrm{D}}_{IC}(K \sqcup K') \wedge K \not\vdash \bot$.

(IC5) holds for Δ^{D}. Let Q_i the elements of $\Delta^{\mathrm{D}}_{IC}(E_1)$, and R_i the elements of $\Delta^{\mathrm{D}}_{IC}(E_2)$. Note that if Q_i is consistent with $K \in E_1$, then $K \subseteq Q_i$ and then if $Q_i \wedge R_j$ is consistent, so $Q_i \wedge R_j \wedge K$ is consistent. So if $K \in E_1$, then $dist_{\mathrm{D}}(Q_i \wedge R_j, K) = dist_{\mathrm{D}}(Q_i, K)$. And similarly for $K \in E_2$, $dist_{\mathrm{D}}(Q_i \wedge R_j, K) = dist_{\mathrm{D}}(R_j, K)$. So if $Q_i \wedge R_j$ is consistent, then $dist_{\mathrm{D}}(Q_i, E_1) + dist_{\mathrm{D}}(R_j, E_2) = dist_{\mathrm{D}}(Q_i \wedge R_j, E_1 \sqcup E_2)$. Since we know that Δ^{C1} satisfies (IC5), we have that $Q_i \wedge R_j \in \mathrm{MAXCONS}(E_1 \sqcup E_2, IC)$. It remains to show that $dist_{\mathrm{D}}(Q_i \wedge R_j, E_1 \sqcup E_2)$ is minimal. If it is not the case $\exists P \in \Delta^{\mathrm{D}}_{IC}(E_1 \sqcup E_2)$ such that $dist_{\mathrm{D}}(P, E_1 \sqcup E_2) < dist_{\mathrm{D}}(Q_i \wedge R_j, E_1 \sqcup E_2)$. So either $dist_{\mathrm{D}}(P, E_1) < dist_{\mathrm{D}}(Q_i \wedge R_j, E_1)$ or $dist_{\mathrm{D}}(P, E_2) < dist_{\mathrm{D}}(Q_i \wedge R_j, E_2)$ hold. Suppose w.l.g. that $dist_{\mathrm{D}}(P, E_1) < dist_{\mathrm{D}}(Q_i \wedge R_j, E_1)$, then $dist_{\mathrm{D}}(P \cap (E_1 \cup IC), E_1) < dist_{\mathrm{D}}(Q_i, E_1)$. So $Q_i \notin \Delta^{\mathrm{D}}_{IC}(E_1)$. Contradiction.

Δ^{D} does not satisfy (IC6) and (IC8). The counterexamples used for Δ^{C1} hold here too.

The proof that (IC7) holds for Δ^{D} is exactly the same that in theorem 2, since add integrity constraints IC_2 does not change the "score" of each maxiconsistent. ■

This operator satisfies as many basic properties as the Δ^{C1} operator. But it satisfies (Maj) instead of (MI), so it can be used to merge knowledge bases, since it takes the distribution of information into account.

Furthermore, the complexity of this operator is not much higher than the one of Δ^{C1} since we only add inclusion tests.

5.2 CARDINALITY OPERATORS

The previous operator is very rough, because the evaluation of a maxiconsistent is a drastic one: a maxiconsistent is ever good or bad for a knowledge base. Therefore, we can expect a more subtle way to evaluate a maxiconsistent. Such a problem has already been addressed in the literature. For example in the case of database update, Fagin *et al.* [FUV83, FKUV86] proposed a notion of *fewer change*:

Definition 12 *Let K_1, K_2 and K be knowledge bases.*

1. *K_1 has fewer insertions than K_2 with respect to K if $K_1 \setminus K \subset K_2 \setminus K$.*
2. *K_1 has fewer deletions than K_2 with respect to K if $K \setminus K_1 \subset K \setminus K_2$.*
3. *K_1 has fewer change than K_2 with respect to K if K_1 has fewer deletions than K_2, or K_1 and K_2 have the same deletions and K_1 has fewer insertions than K_2.*

The problem with Fagin *et al.* definition of fewer change is that it gives only a partial order. Here we need a total order in order to aggregate the "individual" preferences into "social" preferences.

Furthermore, Fagin *et al.* give more importance to deletions than to insertions. Even if it can be justified, for update, by the wish to keep as many as possible of the formulae of the old knowledge, it seems to contradict the "smallest change" requirement, since a set that accomplishes no deletions but adds thousands of formulae, would be considered better than a set that accomplishes one deletion and no insertion. So, from a merging point of view, it seems that we have to give the same importance to deletions as to insertions. This leads to the following operators.

5.2.1 The Symmetrical Difference Operator

The following operator is defined from a distance that denotes the cardinality of the symmetrical difference between the knowledge base and the maxiconsistent sets.

Definition 13 *Consider a knowledge set E and a knowledge base M.*

- $dist_S(M, K) = |K \setminus M| + |M \setminus K|$
- $dist_S(M, E) = \sum_{K \in E} dist_S(M, K)$
- $\Delta_{IC}^{S,\Sigma}(E) = \{M \in \Delta^{C1}{}_{IC}(E) : dist_S(M, E) = \min_{M_i \in \Delta^{C1}{}_{IC}(E)} (dist_S(M_i, E))\}$

So the selected maxiconsitent sets are those that have the least differences (in terms of number of formulae) with the knowledge bases.

The following theorem states the logical properties satisfied by this operator.

Theorem 7 *The $\Delta_{IC}^{S,\Sigma}$ operator satisfies (IC0), (IC1), (IC2), (IC4), (IC7), (IC8) and (Maj). It does not satisfy (IC3), (IC5), (IC6) and (MI).*

Proof: (IC0), (IC1) and (IC2) are directly satisfied by definition.

(IC4) is satisfied by $\Delta^{S,\Sigma}$. It follows from the following property: $\forall M\ K \cap K' \subseteq M \subseteq K \cup K'$ $dist_S(M, K \sqcup K') = |K| + |K'| - 2|K \cap K'|$. Then if $\Delta_{IC}^{S,\Sigma}(K \sqcup K') \wedge K \not\vdash \bot$, that is K is consistent with a maxiconsistent M. It implies that $K \subseteq M$, so by the above property $dist_S(M, K \sqcup K') = |K| + |K'| - 2|K \cap K'|$. But K' is a consistent subset of $K \cup K'$, so there exists an element $M' \in \Delta^{C1}(K \sqcup K')$ such that $K' \subseteq M'$. By the same property we get $dist_S(M', K \sqcup K') = dist_S(M, K \sqcup K')$, so $M' \in \Delta_{IC}^{S,\Sigma}(K \sqcup K')$. So $\Delta_{IC}^{S,\Sigma}(K \sqcup K') \wedge K' \not\vdash \bot$.

(IC5) does not hold for $\Delta^{S,\Sigma}$. Consider the three knowledge bases $K_1 = \{a, a \rightarrow b \wedge c\}$, $K_2 = \{\neg c\}$ and $K_3 = \{a \rightarrow \neg c\}$. And define $E_1 = K_1 \sqcup K_2$, $E_2 = K_3$ and $IC = \top$.

(IC6) does not hold for $\Delta^{S,\Sigma}$. The counterexample for Δ^{C1} holds here too.

(IC7) and (IC8) hold for $\Delta^{S,\Sigma}$. When $\Delta_{IC_1}^{S,\Sigma}(E) \wedge IC_2$ is not consistent (IC7) and (IC8) are satisfied straightforwardly. So assume that $\Delta_{IC_1}^{S,\Sigma}(E) \wedge IC_2$ is consistent. Let P be an element of $\Delta_{IC_1 \wedge IC_2}^{S,\Sigma}(E)$, and let Q be an element of $\Delta_{IC_1}^{S,\Sigma}(E)$ consistent with IC_2. Let's define $A_1 = \bigwedge E \wedge IC_1$ and $A_2 = IC_2 \setminus A_1$. Then we can split $P = P_1 \oplus P_2$ with $P_1 = P \cap A_1$ and $P_2 = P \cap A_2$. Similarly let's define $Q \cup IC_2 = Q_1 \oplus Q_2$ such that $Q_1 = (Q \cup IC_2) \cap A_1$ and $Q_2 = (Q \cup IC_2) \cap A_2$. As $IC_2 \subseteq P$ and $IC_2 \subseteq Q \cup IC_2$, by construction it is easy to see that $P_2 = A_2 = Q_2$. In terms of distances, let $K \in E$, $dist_S(P, K) = dist_S(P_1, K) + |P_2 \setminus K| - |P_2 \cap K|$ and $dist_S(Q \cup IC_2, K) = dist_S(Q_1, K) + |Q_2 \setminus K| - |Q_2 \cap K|$. We have that $|P_2 \setminus K| - |P_2 \cap K| = |Q_2 \setminus K| - |Q_2 \cap K|$. As P is in $\Delta_{IC_1 \wedge IC_2}^{S,\Sigma}(E)$ and $Q \cup IC_2 \subseteq E \cup IC_1 \cup IC_2$, then $dist_S(P, K) \leq dist_S(Q \cup IC_2, K)$. Similarly as $Q_1 = Q$ is in $\Delta_{IC_1}^{S,\Sigma}(E)$ and $P_1 \subseteq E \cup IC_1$, then $dist_S(P_1, K) \geq dist_S(Q, K)$. From this it is easy to see that both $dist_S(P, K) = dist_S(Q \cup IC_2, K)$ and $dist_S(Q, K) = dist_S(P_1, K)$ hold. So $Q \cup IC_2$ is in $\Delta_{IC_1 \wedge IC_2}^{S,\Sigma}(E)$, and P_1 is in $\Delta_{IC_1}^{S,\Sigma}(E)$. ■

This operator doesn't seem to have a lot of logical properties. In particular, it does not satisfy the very impor-

Table 2: Properties of the merging operators based on Δ^{C1}

	IC0	IC1	IC2	IC3	IC4	IC5	IC6	IC7	IC8	MI	Maj
Δ^{C1}	✓	✓	✓	–	✓	✓	–	✓	–	✓	–
Δ^{d}	✓	✓	✓	–	✓	✓	–	✓	–	–	✓
$\Delta^{S,\Sigma}$	✓	✓	✓	–	✓	–	–	✓	✓	–	✓
$\Delta^{\cap,\Sigma}$	✓	✓	✓	–	–	✓	✓	✓	✓	–	✓

tant (IC5) and (IC6) postulates, whereas this is mainly those two postulates that deal with "aggregation" properties of the merging operators.

But instead of focusing on the differences between the knowledge bases and the maxiconsistent sets, we can focus on what is common in those sets. These two approaches are very close in spirit but the second one is more interesting from a logical point of view.

5.2.2 The Intersection Operator

This operator is defined from a distance that denotes the cardinality of the intersection between the knowledge base and the maxiconsistent sets.

Definition 14 *Consider a knowledge set E and a knowledge base M.*

- $dist_{\cap}(M, K) = |K \cap M|$
- $dist_{\cap}(M, E) = \sum_{K \in E} dist_{\cap}(M, K)$
- $\Delta^{\cap,\Sigma}_{IC}(E) = \{M \in \Delta^{C}{}_{IC}(E) : dist_{\cap}(M, E) = \max_{M_i \in \Delta^{C1}{}_{IC}(E)} (dist_{\cap}(M_i, E))\}$

So the selected maxiconsistent sets are those that fit the knowledge bases on a maximum of formulae.

Theorem 8 *The $\Delta^{\cap,\Sigma}_{IC}$ operator satisfies (IC0), (IC1), (IC2), (IC5), (IC6), (IC7), (IC8) and (Maj). It does not satisfy (IC3), (IC4) and (MI).*

Proof: (IC0), (IC1) and (IC2) are straightforwardly satisfied.

(IC4) is not satisfied. Consider the following example: $K = \{a, b\}$ and $K' = \{\neg a \wedge \neg b\}$. Then $\Delta^{\cap,\Sigma}_{\top}(K \sqcup K') = K$ and $\Delta^{\cap,\Sigma}_{\top}(K \sqcup K') \wedge K' \vdash \bot$.

(IC5) holds for $\Delta^{\cap,\Sigma}$. The result is straightforward if $\Delta^{\cap,\Sigma}_{IC}(E_1) \wedge \Delta^{\cap,\Sigma}_{IC}(E_2)$ is not consistent. Otherwise there exists Q_i an element of $\Delta^{\cap,\Sigma}_{IC}(E_1)$ and R_j an element of $\Delta^{\cap,\Sigma}_{IC}(E_2)$ such that $Q_i \wedge R_j$ is consistent. Notice that if $K_l \in E_1$, then $dist_{\cap}(Q_i \wedge R_j, K_l) = dist_{\cap}(Q_i, K_l)$. Because if it is not the case, that is $dist_{\cap}(Q_i \wedge R_j, K_l) > dist_{\cap}(Q_i, K_l)$ then there exists a formula $\alpha \in \bigwedge E_1$ such that $\alpha \notin Q_i$ and $Q_i \cup \alpha$ is consistent. So Q_i is not a maxiconsistent. Contradiction. And similarly if $K_l \in E_2$, then $dist_{\cap}(Q_i \wedge R_j, K_l) = dist_{\cap}(R_j, K_l)$. So if $Q_i \wedge R_j$ is consistent, then $dist_{\cap}(Q_i \wedge R_j, E_1 \sqcup E_2) = dist_{\cap}(Q_i, E_1) + dist_{\cap}(R_j, E_2)$. From properties of Δ^{C1} we know that $Q_i \wedge R_j$ is in $\text{MAXCONS}(E_1 \cup E_2, IC)$. It remains to show that $dist_{\cap}(Q_i \wedge R_j, E)$ is maximum. If it is not the case $\exists P \in \Delta^{\cap,\Sigma}_{IC}(E_1 \sqcup E_2)$ such that $dist_{\cap}(P, E_1 \sqcup E_2) > dist_{\cap}(Q_i \wedge R_j, E_1 \sqcup E_2)$. So either $dist_{\cap}(P, E_1) > dist_{\cap}(Q_i \wedge R_j, E_1)$ or $dist_{\cap}(P, E_2) > dist_{\cap}(Q_i \wedge R_j, E_2)$ hold. Suppose w.l.g. that $dist_{\cap}(P, E_1) > dist_{\cap}(Q_i \wedge R_j, E_1)$, then $dist_{\cap}(P \cap (E_1 \cup IC), E_1) > dist_{\cap}(Q_i, E_1)$. So $Q_i \notin \Delta^{\cap,\Sigma}_{IC}(E_1)$. Contradiction.

(IC6) holds. Let P be an element of $\Delta^{\cap,\Sigma}_{IC}(E_1 \sqcup E_2)$. And decompose $P = P_1 \cup P_2$ with $P_1 = P \cap (IC \cup E_1)$ and $P_2 = P \cap (IC \cup E_2)$. If $P_1 \notin \Delta^{\cap,\Sigma}_{IC}(E_1)$, then exists $Q \in \Delta^{\cap,\Sigma}_{IC}(E_1)$ and $R \in \Delta^{\cap,\Sigma}_{IC}(E_2)$ such that $Q \wedge R$ is consistent and by definition $dist_{\cap}(Q, E_1) > dist_{\cap}(P_1, E_1)$ and $dist_{\cap}(R, E_2) \geq dist_{\cap}(P_2, E_2)$. So with a similar argument than for (IC5) we have that $dist_{\cap}(Q, E_1 \sqcup E_2) > dist_{\cap}(P, E_1 \sqcup E_2)$. So $P \notin \Delta^{\cap,\Sigma}_{IC}(E_1 \sqcup E_2)$. Contradiction.

(IC7) and (IC8) hold. The proof is similar to the one of $\Delta^{S,\Sigma}$. When $\Delta^{\cap,\Sigma}_{IC_1}(E) \wedge IC_2$ is not consistent (IC7) and (IC8) are satisfied straightforwardly. So assume that $\Delta^{\cap,\Sigma}_{IC_1}(E) \wedge IC_2$ is consistent. Let P be an element of $\Delta^{\cap,\Sigma}_{IC_1 \wedge IC_2}(E)$. And let Q be an element of $\Delta^{\cap,\Sigma}_{IC_1}(E)$ consistent with IC_2. Let's define $A_1 = \bigwedge E \cup IC_1$ and $A_2 = IC_2 \setminus A_1$. Then we can split $P = P_1 \oplus P_2$ with $P_1 = P \cap A_1$ and $P_2 = P \cap A_2$. Similarly let's define $Q \cup IC_2 = Q_1 \oplus Q_2$ such that $Q_1 = (Q \cup IC_2) \cap A_1$ and $Q_2 = (Q \cup IC_2) \cap A_2$. As $IC_2 \subseteq P$ and $IC_2 \subseteq Q \cup IC_2$, by construction it is easy to see that $P_2 = A_2 = Q_2$. In terms of distances $dist_{\cap}(P, K) = dist_{\cap}(P_1, K) + dist_{\cap}(P_2, K)$ and $dist_{\cap}(Q, K) = dist_{\cap}(Q_1, K) + dist_{\cap}(Q_2, K)$. But we have that $dist_{\cap}(P_2, K) = dist_{\cap}(Q_2, K)$. As P is in $\Delta^{\cap,\Sigma}_{IC_1 \wedge IC_2}(E)$ and $Q \cup IC_2 \subseteq E \cup IC_1 \cup IC_2$,

then $dist_\cap(P,K) \geq dist_\cap(Q \cup IC_2, K)$. Similarly as $Q_1 = Q$ is in $\Delta^{\cap,\Sigma}_{IC_1}(E)$ and $P_1 \subseteq E \cup IC_1$, then $dist_\cap(P_1,K) \leq dist_\cap(Q,K)$. From this it is easy to see that both $dist_\cap(P,K) = dist_\cap(Q \cup IC_2,K)$ and $dist_\cap(Q,K) = dist_\cap(P_1,K)$ hold. So $Q \cup IC_2$ is in $\Delta^{\cap,\Sigma}_{IC_1 \wedge IC_2}(E)$, and P_1 is in $\Delta^{\cap,\Sigma}_{IC_1}(E)$. ■

The properties of merging operators defined in this section from selection functions are summed up in Table 2. So it is clear that one can get more logical properties with suitable selection functions.

All the operators we have defined in this section satisfy (Maj) (so they do not satisfy (MI)) and therefore are much more satisfactory than Δ^{C1} as merging operators.

We can note, in particular, that the intersection operator $\Delta^{\cap,\Sigma}$ satisfies almost all the properties of IC merging operators. It's very hard for a syntactical operator (*i.e.* for an operator working on knowledge bases that are not closed under logical consequences) to satisfy ($IC3$). So the sole "missing" property is (IC4).

Remark that the only operator that satisfies as many properties as $\Delta^{\cap,\Sigma}$ is the Δ^{C5} operator. But the behaviour of Δ^{C5} is (over-)simpler than the one of $\Delta^{\cap,\Sigma}$. Furthermore the postulate not satisfied by $\Delta^{\cap,\Sigma}$ is (IC4), whereas the one that Δ^{C5} does not satisfy is (IC6). But failing to satisfy (IC6) is worse that not satisfy (IC4) since, in fact, (IC5) and (IC6) are the conditions that purely deal with the aggregation problem. Their semantical counterparts [KP99] are seen as essential conditions for aggregation methods in Social Choice Theory (*cf e.g.* [Sen79]).

Finally, as noted at the beginning of this section, we can apply those methods (or other ones) to the other combination operators (Δ^{C3}, Δ^{C4} and Δ^{C5}) in order to improve their behaviour in the same way.

6 CONCLUSION

We have studied in this paper the logical properties of combination operators. We have shown that, due to the irrelevance of the distribution of information for the combination process, those operators do not have a good behaviour concerning the merging.

Then, we have shown that the use of selection functions can improve the logical properties of combination operators. In particular the intersection operator $\Delta^{\cap,\Sigma}$ satisfies almost all the postulates of IC merging operators.

We have only used a utilitarist aggregation method by adding the different distances when calculating the distance between a maxiconsistent and a knowledge set. So we have only defined majority operators. It could be interesting to see if one obtains similar results with an egalitarian method *à la* leximin (*cf* the Δ^{GMax} operator in [KP99]), leading to arbitration operators.

Another interesting work could be to find the general properties that selection methods have to verify in order to satisfy the postulates, as done in belief revision with *transitively relational partial meet contraction functions* [AGM85, Gär88]. That can give a new representation theorem for IC merging operators.

Acknowledgments

I would like to thank Ramón Pino Pérez for helpful discussions and comments about this work.

References

[AGM85] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

[AM82] C. E. Alchourrón and D. Makinson. The logic of theory change: Contraction functions and their associated revision functions. *Theoria*, 48:14–37, 1982.

[Arr63] K. J. Arrow. *Social choice and individual values.* Wiley, New York, second edition, 1963.

[BCD+93] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, and H. Prade. Inconsistency management and prioritized syntax-based entailment. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI'93)*, pages 640–645, 1993.

[BDL+98] S. Benferhat, D. Dubois, J. Lang, H. Prade, A. Saffioti, and P. Smets. A general approach for inconsistency handling and merging information in prioritized knowledge bases. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 466–477, 1998.

[BDP97] S. Benferhat, D. Dubois, and H. Prade. Some syntactic approaches to the handling of inconsistent knowledge bases: a comparative study, part 1: the flat case. *Studia Logica*, 58:17–45, 1997.

[BKM91] C. Baral, S. Kraus, and J. Minker. Combining multiple knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 3(2):208–220, 1991.

[BKMS92] C. Baral, S. Kraus, J. Minker, and V. S. Subrahmanian. Combining knowledge bases consisting of first-order theories. *Computational Intelligence*, 8(1):45–71, 1992.

[Bre89] G. Brewka. Preferred subtheories: an extended logical framework for default reasoning. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI'89)*, pages 1043–1048, 1989.

[Eth88] D. Etherington. *Reasoning with incomplete information.* Morgan Kaufmann, 1988.

[FKUV86] R. Fagin, G. M. Kuper, J. D. Ullman, and M. Y. Vardi. Updating logical databases. *Advances in Computing Research*, 3:1–18, 1986.

[FUV83] R. Fagin, J. D. Ullman, and M. Y. Vardi. On the semantics of updates in databases. In *Proceedings of the 2nd ACM SIGACT-SIGMOD Symposium on the Principles of Database Systems*, pages 352–365, 1983.

[Gär88] P. Gärdenfors. *Knowledge in flux.* MIT Press, 1988.

[HM92] J. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, 1992.

[KM91] H. Katsuno and A. O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52:263–294, 1991.

[KP98] S. Konieczny and R. Pino Pérez. On the logic of merging. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 488–498, 1998.

[KP99] S. Konieczny and R. Pino Pérez. Merging with integrity constraints. In *Proceedings of the Fifth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'99)*, LNAI 1638, pages 233–244, 1999.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Sen79] A. K. Sen. *Collective Choice and Social Welfare.* Advanced Textbooks in Economics. Elsevier, 1979.

BReLS: A System for the Integration of Knowledge Bases

Paolo Liberatore and **Marco Schaerf**
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198, Roma, Italy
Email: liberato@dis.uniroma1.it schaerf@dis.uniroma1.it

Abstract

The process of integrating knowledge coming from different sources has been widely investigated in the literature. Three distinct conceptual approaches to this problem have been most succesful: belief revision, merging and update.

In this paper we present a framework that integrates these three approaches. In the proposed framework all three operations can be performed. We provide an example that can only be solved by applying more than one single style of knowledge integration and, therefore, cannot be addressed by anyone of the approaches alone.

The framework has been implemented, and the examples shown in this paper (as well as other examples from the belief revision literature) have been successfully tested.

1 Introduction

In this paper we introduce a new framework for the integration of information coming from different sources. The proposed framework allows for the formalization of complex domains where the information can have different degrees of reliability and can arrive at different time points.

Many formalisms for the integration of knowledge bases have been introduced in the literature in the last years. Among the most relevant ones we have belief revision theory, due to Alchourrón, Gärdenfors, and Makinson [1] and update, discussed by Katsuno and Mendelzon [7].

In the original formulation of the belief revision theory, due to Alchourrón, Gärdenfors, and Makinson [1], only a revision operator is considered. Revision has been initially defined as the operation that allows modifying our knowledge referring to a static world (i.e. a scenario that does not change) when some new piece of information comes to be known. The work of Winslett [17] made clear that when information is about a scenario that may change, the expected properties of the integration operator are different. For this reason, the operator of revision has in this case a different name: update. A detailed discussion of the differences between revision and update has been carried on by Katsuno and Mendelzon [6].

Both revision and update assume that the new piece of information is correct. This assumption must be removed if we want to merge two pieces of information, referring the same time point, that have the same degree of reliability. This is a very common scenario: we have information coming from difference sources, with the same degree of reliability, that may contradict each other. The case in which two sources contradict each other is indeed the most interesting one (if all pieces of information can be put together without obtaining an inconsistency, we can simply do it). There is no consensus, in the literature, about the name to give to the operation of combining knowledge bases with the same degree of reliability: Revesz [15] and Liberatore and Schaerf [10, 11] use the name *arbitration*, while Lin and Mendelzon [13] and Lin [12] prefer the term *merging*. Konieczny and Pino Perez [8] show that there is a semantical distinction between arbitration and merging. Namely, they show that there are scenarios in which arbitration is more appropriate, and other ones in which the operation to perform is merging. This issue is relatively new, so there is no agreement on the semantics yet.

Summarizing, the three operators for knowledge integration can be described as follows.

Revision: integration of two pieces of information, in

which one is considered fully reliable, while the other one may be partially incorrect;

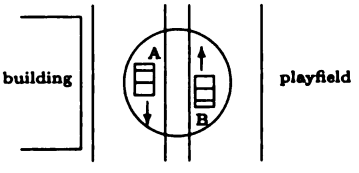
Update: integration of two pieces of information, both fully reliable, referring to two different time point (some changes may have occurred in between);

Merging/Arbitration: integration of two (or more) pieces of information having the same degree of reliability.

The framework we present allows for all of the above forms of reasoning to be performed. We now present an example that calls for the use of both revision and merging.

Example 1 *Let us consider the following scenario: we have a system to control speed and direction of a car. In order to decide what is the current position of the car we have four sources of information: the estimated position of the car using some satellite system; the information coming from sonar sensors mounted on the car (they can percept the presence of buildings along the road); the vision system, which can perform the same operation; and finally, we have the map of the city.*

Suppose that the satellite indicates the position of the car to be inside the circle in the following picture:



As a result, the car may be either in the position marked A, or in the position marked B. In logical terms, this can be represented as $\mathtt{pos}_A \not\equiv \mathtt{pos}_B$, *i.e. the car is either in position A or in position B (but not both). From the map we know that if the car is in position A, there should be a building on its right side, and if it is in position B, there should be a building on its left side. As a result, we have* $\mathtt{pos}_A \rightarrow \neg\mathtt{building\_left} \wedge \mathtt{building\_right}$, *and* $\mathtt{pos}_B \rightarrow \mathtt{building\_left} \wedge \neg\mathtt{building\_right}$.

The sensors say that there is a building on the left, and nothing on the right. This piece of information can be represented as $\mathtt{building\_left} \wedge \neg\mathtt{building\_right}$. *The information coming from the vision system is that there are buildings both on the left and on the right. In propositional terms, this can be represented as* $\mathtt{building\_left} \wedge \mathtt{building\_right}$.

In this example, there is a pair of sources of information which are fully reliable, that is, the position as given by the satellite and the map of the city. In this case, these pieces of information are consistent with each other, but they are not complete. What we know is that the car is in one of two possible positions. Since we want the exact position of the car, we should use the information coming from the two other sources, which are less reliable.

The problem can be solved by observing that, if the car is in position A, we have to assume that both the sensors and the vision system are faulty. On the other hand, the car being in position B means that only the vision system is wrong. As a result, this second possibility should be preferred.

In this example, we have both revision and merging, as there is a set of sources of information, and some of them have the same degree of reliability. Note that, in more complex scenarios, there may be more than two sources, all having a different degree of reliability (e.g. three sources each having a degree of reliability different from the other ones).

Let now consider an example in which we have more than one time point, but revision is also needed because we have a contradiction that cannot be solved by assuming that something changed.

Example 2 *We consider a client-server architecture. A client computer sends requests of service to the server; the server receives them and executes some kind of computation, and immediately sends a message back to the client informing it that the request has been processed. The server may not execute the requests either because it is too busy (and then the requests are delayed) or it is crashed (in this case, requests are never executed).*

Suppose that, after sending a service request, the client receives no answer. After some time, it assumes that the server is crashed, and thus the request has not been processed. Expressing such a knowledge in the propositional language, we have, at time 1, the following pieces of knowledge: $\neg\mathtt{service\_executed}$ *and* $\mathtt{crashed}$.

Eventually, the client receives an answer from the server, stating that the service has been executed: this also implies that the server is not crashed. Now, let us consider how reliable our knowledge is:

$\neg\mathtt{service\_executed}$ at time 1. *Since the client did not receive any acknowledgment of execution, we can believe that the server did not process the request (this is because the server sends the answer*

immediately after processing the request and we assume that the server does not crash after the execution but before sending the answer).

crashed at time 1. *This is an assumption made by the client. Indeed, if the answer is late from the server, the client first believe that the server is busy, and only after some time it starts to think that it crashed.*

service_executed $\wedge \neg$**crashed** at time 2.
We have received the answer, thus the service has been executed, and the server is not crashed. This piece of information is fully reliable.

Now, we have also some assumptions on change: a server cannot restart working after a crash. It is quite clear what happened in this scenario. At time 1 the server was not crashed: it was simply forced to delay the processing of the request. The processing was executed between time 1 and 2.

In the scenario of the example both revision and update are necessary. Indeed, the initial knowledge (that is, the knowledge at time 1) was that the server was crashed and the service was not executed. We have then a revision of this information, because we discover that the server was not crashed, after all. We also have an update, since the execution of the request happened between time 1 and 2.

In order to help experimenting with our system, we have implemented BReLS. The current implementation is available at the URL
`http://www.dis.uniroma1.it/~liberato/brels`
A CGI interface allows the system to be run over the Internet.

2 Syntax

In this section we describe the syntax of BReLS. As shown in Example 1, different pieces of information may have different degrees of reliability. In BReLS, we write:

$$\texttt{source}(i) : K$$

In words, we mean that the piece of knowledge K has reliability i. Knowledge is expressed using propositional formulas, while reliability is given as positive integer numbers. Thus, K must be a propositional formula and i an integer number such that $i > 0$. Our assumption is that the satellite system and the map always provide fully reliable information, while the sensor and the vision systems may be wrong. This can be formalized by assigning an higher degree of reliability to the information provided by the satellite and inferred from the map. As a result, Example 1 can be encoded as follows:

$$\begin{array}{ll}
\texttt{source}(2): & \texttt{pos}_A \not\equiv \texttt{pos}_B \\
\texttt{source}(2): & (\texttt{pos}_A \rightarrow \neg\texttt{building_left} \wedge \texttt{building_right}) \\
 & \wedge(\texttt{pos}_B \rightarrow \texttt{building_left} \wedge \neg\texttt{building_right}) \\
\texttt{source}(1): & \texttt{building_left} \wedge \neg\texttt{building_right} \\
\texttt{source}(1): & \texttt{building_left} \wedge \texttt{building_right}
\end{array}$$

In BReLS, information having degree of reliability i is always considered more reliable than any number of pieces of information with reliability $j < i$. In other words, a source saying that a variable is true is always believed, even if there are hundreds of less reliable sources saying that the variable is false.

Let us now introduce time in our framework. Borrowing Sandewall's syntax [16], we write time in square brackets:

$$\texttt{source}(i) : [t]K$$

This formula means that the information K is believed to hold at time t with a degree of reliability i. When time is not specified, we assume by default time point 1. Note that we assume that time is discrete and linear.

The fact that scenarios may be modified is formalized using the idea of *changes*. A change is what in the reasoning about actions field is known as action, and in belief revision as events [3]. Essentially, a change is something that modifies the world of interest. The default assumption is that changes do not happen. However, we can explicitly say that a change may happen with a given "penalty". The general form of change statements is:

$$\texttt{change(i)}: \quad [t]l$$

In this statement, i is a nonnegative integer, while l is a literal. It means that l may become true, and the penalty associated to this event is given by the integer i. The greater the value of the penalty i, more unlikely is the change. In most cases, the penalty of changes is independent from time. In this case, we have the simplified form:

$$\texttt{change(i)}: \quad l$$

which means that the penalty of the change making l true is i for any time point. We have also two ways

for expressing the penalty of changes when this is independent of the literal:

$$\texttt{change(i)}: \quad [t]$$
$$\texttt{change(i)}$$

The first statement denotes that any change at time t has penalty i, while the second one states that any change at any time point has penalty i. When the change statements are inconsistent, we assume that the most specific one holds. For instance, if we have $\{\texttt{change(1)}, \texttt{change(2)} : l\}$ then all changes have penalty 1, except l which has penalty 2. More precisely, the penalty of the change that makes l true between time points t and $t+1$ is specified as follows:

1. If there is a statement $\texttt{change(i)} : [t]l$, the penalty is i, regardless of the other change statements;
2. Else if the knowledge base contains a change statement $\texttt{change(j)} : [t]$, then j is the penalty;
3. Else if there is a change statement $\texttt{change(k)} : l$, then the penalty is k;
4. Else if there exists a change statement $\texttt{change(m)}$, then the penalty is m;
5. Else the penalty is 1;

Let us now consider how Example 2 can be expressed using the syntax of BReLS. The knowledge about time points can be simply expressed as:

$\texttt{source(2)}: \quad [1]\neg\texttt{service\_executed}$
$\texttt{source(1)}: \quad [1]\texttt{server\_crashed}$
$\texttt{source(2)}: \quad [2]\texttt{service\_executed} \wedge \neg\texttt{server\_crashed}$

Indeed, the pieces of knowledge with reliability 2 are those expressing things known for certain. On the other hand, [1]server_crashed is just an assumption made, thus it is less reliable. Now, any change may happen, except that a server cannot recover from a crash. As a result, we give penalty 1 to any change except that in which the server becomes non-crashed.

$\texttt{change(1)}$
$\texttt{change(4)}: \quad \texttt{server\_crashed}$

The first statement says that all changes have penalty 1 (where not otherwise specified), thus they may happen. The second statement says that the variable server_crashed cannot switch from false to true. By default, the statement change(1) can be omitted, that is, every change whose penalty is not given is assumed to have penalty 1. A penalty 0 for a change means that there is no problem in assuming that the change happened, that is, we do not make the default assumption that the change does not happen.

3 Semantics

In this section we first introduce the notion of dynamic model, and then we provide two different semantics.

3.1 Models

Suppose the time points we are dealing with are $\{1, \ldots, T\}$, and the propositional variables are $X = \{x_1, \ldots, x_n\}$.

Definition 1 *A* static model *is a truth assignment to the variables in X.*

A static model represents the state of the world in a given time point. In order to express the state and the changes in all considered time points, we introduce the notion of dynamic model.

Definition 2 *A* dynamic model *is composed of two parts:*

1. *an initial (static) model M_0, for the time point 0;*
2. *a sequence of (possibly empty) sets of changes $C_1, \ldots C_n$ for each pair of consecutive time points.*

This is a very general definition, without any explicit definition of what changes are. In BReLS, the only changes considered are those setting the value of a variable. Changes are represented by literals: for instance the change that makes a false is represented by $\neg a$. Given M_0 and the sequence $C_1, \ldots C_n$ we can unambigously define the static model at time t as the result of iteratively applying the changes in $C_1 \ldots C_t$ to the initial model M_0.

3.2 Preference of Static Models.

Let M be a static model, and F a propositional formulas. We define the distance between M and F as:

$$\text{dist}(M, F) = \min(\{\text{hamm}(M, N) | N \in \text{mod}(F)\}, \leq)$$

where $\text{hamm}(M, N)$ is the Hamming distance between the models M and N (the number of atoms to which they assign a different truth value). In words: the distance between a static model M and a propositional

formula F is the minimal Hamming distance between M and any model of F.

Let us consider a knowledge base composed of two formulas with the same reliability and referring to the same time point:

$$\begin{aligned} \texttt{source(1)}: &\quad K_1 \\ \texttt{source(1)}: &\quad K_2 \end{aligned}$$

Since we prefer models that are as close as possible to the formulas in the knowledge base, we define the preference of a static model M as:

$$\text{pref}(M) = \text{dist}(M, K_1) + \text{dist}(M, K_2)$$

The models of a knowledge base are those with the minimal value of the preference function, thus the models in which the sum of the distances is minimal. Let us now consider a knowledge base in which there is another formula with a greater degree of reliability, for instance:

$$\begin{aligned} \texttt{source(1)}: &\quad K_1 \\ \texttt{source(1)}: &\quad K_2 \\ \texttt{source(2)}: &\quad K_3 \end{aligned}$$

Since K_3 is the most reliable formula, all minimal models should satisfy it, thus the preference of a model cannot be defined as a sum of distances. The preference of a model is instead defined as an array having one element for each level of reliability in the knowledge base. In our example, there are two formulas with reliability 1 and one formula with reliability 2, thus we need an array with two elements. The array is defined as:

$$\text{pref}(M)[i] = \sum_{F \text{ has reliability } i} \text{dist}(M, F)$$

In the specific example above, the array has two elements, whose value is:

$$\begin{aligned} \text{pref}(M)[1] &= \text{dist}(M, K_1) + \text{dist}(M, K_2) \\ \text{pref}(M)[2] &= \text{dist}(M, K_3) \end{aligned}$$

The ordering between two models M_1 and M_2 is defined as follows, where P is the maximal level of reliability of formulas in the knowledge base.

$$\begin{aligned} M_1 \prec M_2 \quad \text{iff} \quad & \exists i \ (1 \leq i \leq P) \text{ such that} \\ & (\text{pref}(M_1)[i] < \text{pref}(M_2)[i]) \text{ AND} \\ & \forall j. \ (i < j \leq P) \text{ implies} \\ & (\text{pref}(M_1)[j] = \text{pref}(M_2)[j]) \end{aligned}$$

This definition ensures that if the distance of M_1 to the formulas with the maximal reliability is less than the distance of M_2, then M_1 is preferred, regardless of the formulas having lower reliability. The distance to formulas with reliability $P-1$ only matters when the distance to formulas with reliability P is the same for both models.

As a result, if there is only one formula with the greatest reliability, then all minimal models of the knowledge base have static models satisfying that formula (if consistent). This is what is expressed by the AGM postulate of success: if a formula with the greatest priority is consistent, then all models of the revised knowledge base must satisfy it. On the other hand, our ordering satisfies the principle that if a formula with a lower reliability is consistent with the ones with greatest priority, the result of revision is the logical conjunction of them.

3.3 Preference between Dynamic Models

So far, we have defined the array of preference of a static model with respect to a set of formulas referring to the same time point. We now consider knowledge bases with time. In order to define the preference between dynamic models, we have to take into account two facts: 1. there are formulas about different time points, and 2. there are penalties associated with changes.

Let KB be a knowledge base, and let $KB(t,p)$ be the set of its formulas with time t and priority p. Let $DM(t)$ be the static model associated with time t in the dynamic model DM. The preference array of a dynamic model DM is defined as follows.

$$\text{pref}(DM)[p] = DM_p + \sum_{t=1}^{T} \sum_{F \in KB(t,p)} \text{dist}(DM(t), F)$$

where DM_p is the number of changes with penalty p in the model DM. The definition of $\prec$ is exactly as in the case of static models. When all formulas are fully reliable (that is, when their degree of priority is greater than those of changes), this definition can be rephrased as: consider the minimal set of changes needed to explain the observations. This is similar to the principle of "actions to explain observation", which is studied in reasoning about actions [9, 2, 14].

There are two possible ways for defining the models of a knowledge base. We call these two possible semantics *backward* and *pointwise*. The first one is the simplest to explain, and allows for expressing scenarios in which knowledge at later time may rule out some

initial models. The second one is based on the principle of "pointwiseness": each initial model should be updated separately (this is one of the basic principles of update according to Katsuno and Mendelzon [6]).

Backward Semantics. We define the ordering $\prec$ between dynamic models from the arrays of preference, as we did for static models. The models of a knowledge base are the minimal dynamic models.

$$\mathrm{mod}(KB) = \min(\{DM\}, \prec)$$

Pointwise Semantics. This is the case in which update should be pointwise, that is, there should be a different set of minimal (dynamic) models for each minimal initial (static) model. The set of models of a knowledge base is the set of minimal models according to the ordering $\trianglelefteq$, defined as:

$$DM \trianglelefteq DM' \quad \text{iff} \quad DM(1) = DM'(1) \text{ and } DM \preceq DM'$$

In other words, given the ordering $\trianglelefteq$ above, the models of KB are defined as:

$$\mathrm{mod}(KB) = \min(\{DM\}, \trianglelefteq)$$

This definition can be reformulated as follows:

$$\mathrm{mod}(KB) = \bigcup_{M \text{ minimal initial model}} \min(\{DM \mid DM(1) = M\}, \prec)$$

This way, for each minimal initial (static) model M, we take the set of minimal dynamic models such that $DM(1) = M$. This equivalent formulation shows that the pointwise semantics obeys the principle that each initial model should be updated separately, which is taken by Katsuno and Mendelzon [6] as a basic principle of update.

Logical entailment under these semantics is defined in a classical fashion. More precisely, a knowledge base KB implies a formula $[t]Q$ under the backward (pointwise) semantics, denoted by $KB \models_{\preceq} [t]Q$, if and only if Q is true at time t in any minimal, under the $\preceq$ ($\trianglelefteq$) ordering, model of the knowledge base.

4 Expressing Revision, Update and Merging

In this section, we show how revision, merging, and update can be encoded in our framework. Namely, we show how "classical" revision can be performed using BReLS, and then we show the same for update. We remark, however, that these operators can be considered as *restrictions* of the full semantics. What is really new in BReLS is the fact that they can be performed together on the same knowledge base.

We can express belief revision in BReLS as follows: First, K and P hold at the same time point, and we have more confidence in P. As a result, we have:

$$KB = \left\{ \begin{array}{ll} \texttt{source}(1): & K, \\ \texttt{source}(2): & P \end{array} \right\}$$

In BReLS, statements without an explicit time point specification are assumed to refer to time 1.

What in the AGM framework is denoted as $K * P$ is actually the formula representing our knowledge after the integration of K and P. As a result, we can say that the result of revising K with P is given by:

$$\mathrm{mod}(K * P) = \{DM(1) \mid DM \text{ is a model of } KB\}$$

In other words, encoding K and P in BReLS according to the principles of revision, we obtain the revision operator $*$ defined by the above formula. Note that all statements refer to the same time point, hence, there is no difference in using the backward or pointwise semantics. It is quite easy to prove that $*$ is indeed a revision operator satisfying all AGM postulates for revision.

Let us now consider update. The update operator is used when we have two pieces of knowledge referring to two different time points. In the initial formulation of update [17] these two pieces of knowledge are considered to be fully reliable. However, they can contradict each other. In order to explain the contradiction, we assume that something change in the world.

The most evident difference between revision and update is that, when the two involved pieces of knowledge are consistent to each other, the result of revision is always the union (conjunction) of them, while the result of update may be different [6].

Let K be the piece of knowledge referring to the first time point, and P the other one. Assume that the first time point is 1, and the second one is 2. The assumptions we made can be encoded in BReLS as follows:

$$KB = \left\{ \begin{array}{ll} \texttt{source}(2): & [1]K, \\ \texttt{source}(2): & [2]P, \\ \texttt{change}(1) & \end{array} \right\}$$

In words, K holds at time 1 while P holds at time 2. Moreover, both pieces of knowledge are reliable, in the sense that models with changes are always preferred

over models that do not satisfy K or P. The result of the update is defined as the formula representing the static models referring to time 2, that is,

$$\mathrm{mod}(K \circ P) = \{DM(2) \mid DM \text{ is a model of } KB \text{ using the pointwise semantics } \}$$

It can be easily proved that $\circ$ is equivalent to Forbus' update: for any pair of formulas K and P it holds $K \circ P \equiv K \circ_F P$, where $\circ_F$ is the update defined by Forbus [4].

Note, however, that basic update operators like Forbus' one suffer from some drawbacks. For example, they do not allow revision of initial state. In other words, while knowledge about time 1 gives some information about time 2, the converse does not hold: information about time 2 does not extend our knowledge about the previous time point. The litmus test example [3] shows a scenario in which such backward inference should be done.

Let us now consider merging, or arbitration. This is the operation of integrating two knowledge bases having the same degree of reliability [15, 13, 10, 12, 11, 8]. The assumption is that the two knowledge bases refer to the same time point. Let the two knowledge bases be represented by the formulas K_1 and K_2. It is straightforward to express them in BReLS:

$$KB = \left\{ \begin{array}{ll} \texttt{source(1)}: & K_1, \\ \texttt{source(1)}: & K_2 \end{array} \right\}$$

The result of arbitration or merging is the formula representing the result of the integration of the two pieces of information. The arbitration operator defined by encoding K_1 and K_2 as above is thus:

$$\mathrm{mod}(K_1 \triangle K_2) = \{DM(1) \mid DM \text{ is a model of } KB\}$$

Let us now consider the properties of this operator. First of all, it is commutative, in the sense that $K_1 \triangle K_2 \equiv K_2 \triangle K_1$. Second, if K_1 and K_2 are consistent to each other, we have $K_1 \triangle K_2 \equiv K_1 \wedge K_2$. Moreover, $K_1 \triangle K_2$ is always consistent. Finally, $K_1 \triangle K_2$ only depends on the sets of models of K_1 and K_2, that is, $\triangle$ is a syntax-independent operator. Since these are the four basic property of merging, we can conclude that the way BReLS integrates knowledge bases obeys the basic principles of merging.

On the negative side, $\triangle$ does not fulfill the basic property of arbitration. Namely, it does not hold $\mathrm{mod}(K_1 \triangle K_2) \subseteq \mathrm{mod}(K_1) \cup \mathrm{mod}(K_2)$. This means that we can perform merging, but not arbitration. However, this property can be enforced using a slightly different encoding:

$$KB' = \left\{ \begin{array}{ll} \texttt{source(2)}: & K_1 \vee K_2, \\ \texttt{source(1)}: & K_1, \\ \texttt{source(1)}: & K_2 \end{array} \right\}$$

and then defining ∇ like $\triangle$:

$$\mathrm{mod}(K_1 \nabla K_2) = \{DM(1) \mid DM \text{ is a model of } KB'\}$$

Since the ∇ operator has the same four properties of $\triangle$ and it holds $\mathrm{mod}(K_1 \nabla K_2) \subseteq \mathrm{mod}(K_1) \cup \mathrm{mod}(K_2)$, we conclude that ∇ is a valid arbitration operator. We note also that it is trivial to extend both $\triangle$ and ∇ in order to integrate more than two knowledge bases with the same degree of reliability.

5 Remarks and Conclusions

As a final step in our analysis we charcterize the complexity of the two semantics of BReLS. The problem we analyze is the problem of entailment, that is, given a knowledge base KB and a formula $[t]Q$, decide whether KB implies $[t]Q$. The results are:

- Deciding whether $KB \models [t]Q$ holds under the pointwise semantics is Π_2^p-complete.
- Deciding whether $KB \models [t]Q$ under the backward semantics is Δ_2^p-complete.

These are good news, since the complexity of reasoning in BReLS is not higher than the complexity of reasoning in other, simpler, systems for belief revision where priorities and time cannot be expressed.

Summing up our contribution, in this paper we have shown how revision, merging and update can be implemented in a system able to perform all of them on the same knowledge base.

The specific choice of the ordering of models in BReLS is fixed. While we can deal with many example using this semantics, there are examples that cannot be correctly dealt by BReLS. For instance, while we can express a problem similar to the Stolen Car Example [5], we cannot express the problem itself, because there is no way to state that the mileage of a car can increase only if the car is outside the parking lot. More work is needed to fully understand how the semantics can be extended to deal with such scenarios.

Further work is also needed in the analysis of more efficient algorithms. Indeed, in its current implementation, BReLS can deal only with small instances. For

this reason, if we want to really use it in real-world domains, efficient algorithms should be developed and implemented.

Acknowledgements

Part of the work reported in this paper has been done while the first author was visiting Linköping University. He thanks Erik Sandewall for his hospitality and support. This work has been partially supported by the Italian Space Agency (ASI).

References

[1] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

[2] C. Baral, M. Gelfond, and A. Provetti. Representing actions: Laws, observations and hypothesis. *Journal of Logic Programming*, 1996.

[3] C. Boutilier. Generalized update: belief change in dynamic settings. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1550–1556, 1995.

[4] K. D. Forbus. Introducing actions into qualitative simulation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI'89)*, pages 1273–1278, 1989.

[5] N. Friedman and J. Y. Halpern. A knowledge-based framework for belief change: Part II: Revision and update. In *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR'94)*, pages 190–200, 1994.

[6] H. Katsuno and A. O. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR'91)*, pages 387–394, 1991.

[7] H. Katsuno and A. O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52:263–294, 1991.

[8] S. Konieczny and R. Pino Perez. On the logic of merging. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 488–498, 1998.

[9] R. Li and L. Pereira. What is believed is what is explained. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI'96)*, pages 550–555, 1996.

[10] P. Liberatore and M. Schaerf. Arbitration: A commutative operator for belief revision. In *Proceedings of the Second World Conference on the Fundamentals of Artificial Intelligence (WOCFAI'95)*, pages 217–228, 1995.

[11] P. Liberatore and M. Schaerf. Arbitration (or how to merge knowledge bases). *IEEE Transactions on Knowledge and Data Engineering*, 10(1):76–90, 1998.

[12] J. Lin. Integration of Weighted Knowledge Bases. *Artificial Intelligence*, 83(2):363–378, 1996.

[13] J. Lin and A. Mendelzon. Knowledge base merging by majority. Manuscript, 1994.

[14] S. McIlraith. Explanatory diagnosis: Conjecturing actions to explain observations. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 167–179, 1998.

[15] P. Z. Revesz. On the semantics of theory change: Arbitration between old and new information. In *Proceedings of the Twelfth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'93)*, pages 71–82, 1993.

[16] E. Sandewall. *Features and Fluents.* Oxford University Press, 1994.

[17] M. Winslett. *Updating Logical Databases.* Cambridge University Press, 1990.

Representing and Aggregating Conflicting Beliefs

Pedrito Maynard-Reid II
Department of Computer Science
Stanford University
Stanford, CA 94305, USA
pedmayn@cs.stanford.edu

Daniel Lehmann
School of Computer Science and Engineering
Hebrew University
Jerusalem 91904, Israel
lehmann@cs.huji.ac.il

Abstract

We consider the two-fold problem of representing collective beliefs and aggregating these beliefs. We propose modular, transitive relations for collective beliefs. They allow us to represent conflicting opinions and they have a clear semantics. We compare them with the quasi-transitive relations often used in Social Choice. Then, we describe a way to construct the belief state of an agent informed by a set of sources of varying degrees of reliability. This construction circumvents Arrow's Impossibility Theorem in a satisfactory manner. Finally, we give a simple set-theory-based operator for combining the information of multiple agents. We show that this operator satisfies the desirable invariants of idempotence, commutativity, and associativity, and, thus, is well-behaved when iterated, and we describe a computationally effective way of computing the resulting belief state.

Keywords: representation of beliefs, multi-agent systems

1 Introduction

We are interested in the multi-agent setting where agents are informed by sources of varying levels of reliability, and where agents can iteratively combine their belief states. This setting introduces three problems: (1) Finding an appropriate representation for collective beliefs; (2) Constructing an agent's belief state by aggregating the information from informant sources, accounting for the relative reliability of these sources; and, (3) Combining the information of multiple agents in a manner that is well-behaved under iteration.

The Social Choice community has dealt extensively with the first problem (although in the context of representing collective preferences rather than beliefs) (cf. (Sen 1986)). The classical approach has been to use quasi-transitive relations (of which total pre-orders are a special subclass) over the set of possible worlds. However, these relations do not distinguish between group indifference and group conflict, and this distinction can be crucial. Consider, for example, a situation in which all members of a group are indifferent between movie a and movie b. If some passerby expresses a preference for a, the group may very well choose to adopt this opinion for the group and borrow a. However, if the group was already divided over the relative merits of a and b, we would be wise to hesitate before choosing one over the other just because a new supporter of a appears on the scene. We propose a representation in which the distinction is explicit. We also argue that our representation solves some of the unpleasant semantical problems suffered by the earlier approach.

The second problem addresses how an agent should actually go about combining the information received from a set of sources to create a belief state. Such a mechanism should favor the opinions held by more reliable sources, yet allow less reliable sources to voice opinions when higher ranked sources have no opinion. True, under some circumstances it would not be advisable for an opinion from a less reliable source to override the agnosticism of a more reliable source, but often it is better to accept these opinions as default assumptions until better information is available. (Maynard-Reid II and Shoham 2000) provides a solution to this problem when belief states are represented as total pre-orders, but runs into Arrow's Impossibility Theorem (Arrow 1963) when there are sources of equal reliability. As we shall see, the generalized representation allows us to circumvent this limitation.

To motivate the third problem, consider the follow-

ing dynamic scenario: A robot controlling a ship in space receives from a number of communication centers on Earth information about the status of its environment and tasks. Each center receives information from a group of sources of varying credibility or accuracy (e.g., nearby satellites and experts) and aggregates it. Timeliness of decision-making in space is often crucial, so we do not want the robot to have to wait while each center sends its information to some central location for it to be first combined before being forwarded to the robot. Instead, each center sends its aggregated information directly to the robot. Not only does this scheme reduce dead time, it also allows for "anytime" behavior on the robot's part: the robot incorporates new information as it arrives and makes the best decisions it can with whatever information it has at any given point. This distributed approach is also more robust since the degradation in performance is much more graceful should information from individual centers get lost or delayed.

In such a scenario, the robot needs a mechanism for combining or *fusing* the belief states of multiple agents potentially arriving at different times. Moreover, the belief state output by the mechanism should be invariant with respect to the order of agent arrivals. We will describe such a mechanism.

The paper is organized as follows: After some preliminary definitions and a discussion of the approach to aggregation taken in classical Social Choice, we introduce modular, transitive relations for representing generalized belief states. We then describe how to construct the belief state of an agent given the belief states of its informant sources when these sources are totally pre-ordered. Finally, we describe a simple set-theory-based operator for fusing agent belief states that satisfies the desirable invariants of idempotence, commutativity, and associativity, and we describe a computationally effective way of computing this belief state.

2 Preliminaries

We begin by defining various well-known properties of binary relations[1]; they will be useful to us throughout the paper.

Definition 1 *Suppose $\leq$ is a relation over a finite set Ω, i.e., $\leq \subseteq \Omega \times \Omega$. We shall use $x \leq y$ to denote $(x, y) \in \leq$ and $x \not\leq y$ to denote $(x, y) \notin \leq$. The relation $\leq$ is:*

1. reflexive *iff* $x \leq x$ *for* $x \in \Omega$. *It is* irreflexive *iff* $x \not\leq x$ *for* $x \in \Omega$.
2. symmetric *iff* $x \leq y \Rightarrow y \leq x$ *for* $x, y \in \Omega$. *It is* asymmetric *iff* $x \leq y \Rightarrow y \not\leq x$ *for* $x, y \in \Omega$. *It is* anti-symmetric *iff* $x \leq y \wedge y \leq x \Rightarrow x = y$ *for* $x, y \in \Omega$.
3. *the* strict version *of a relation* $\leq'$ *over* Ω *iff* $x \leq y \Leftrightarrow x \leq' y \wedge y \not\leq' x$ *for* $x, y \in \Omega$.
4. total *iff* $x \leq y \vee y \leq x$ *for* $x, y \in \Omega$.
5. modular *iff* $x \leq y \Rightarrow x \leq z \vee z \leq y$ *for* $x, y, z \in \Omega$.
6. transitive *iff* $x \leq y \wedge y \leq z \Rightarrow x \leq z$ *for* $x, y, z \in \Omega$.
7. quasi-transitive *iff its strict version is transitive.*
8. *the* transitive closure *of a relation* $\leq'$ *over* Ω *iff* $x \leq y \Leftrightarrow \exists w_0, \ldots, w_n \in \Omega.\ x = w_0 \leq' \cdots \leq' w_n = y$ *for some integer* n, *for* $x, y \in \Omega$.
9. acyclic *iff* $\forall w_0, \ldots, w_n \in \Omega.\ w_0 < \cdots < w_n$ *implies* $w_n \not< w_0$ *for all integers* n, *where* $<$ *is the strict version of* $\leq$.
10. *a* total pre-order *iff it is total and transitive. It is a* total order *iff it is also anti-symmetric.*
11. *an* equivalence relation *iff it is reflexive, symmetric, and transitive.*

Proposition 1

1. *The transitive closure of a modular relation is modular.*[2]
2. *Every transitive relation is quasi-transitive.*
3. *(Sen 1986) Every quasi-transitive relation is acyclic.*

Given a relation over a set of alternatives and a subset of these alternatives, we often want to pick the subset's "best" elements with respect to the relation. We define this set of "best" elements to be the subset's *choice set*:

Definition 2 *If $\leq$ is a relation over a finite set Ω, $<$ is its strict version, and $X \subseteq \Omega$, then the* choice set of X with respect to $\leq$ *is*

$$C(X, \leq) = \{x \in X : \nexists x' \in X.\ x' < x\}.$$

[1] We only use binary relations in this paper, so we will refer to them simply as relations.

[2] Due to space considerations, we have omitted all proofs from this manuscript. They can be found at the website `http://robotics.stanford.edu/~pedmayn/Papers/`.

A *choice function* is one which assigns to every subset X a non-empty subset of X:

Definition 3 *A* choice function *over a finite set Ω is a function $f : 2^\Omega \setminus \emptyset \to 2^\Omega \setminus \emptyset$ such that $f(X) \subseteq X$ for every $X \subseteq \Omega$.*

Now, every acyclic relation defines a choice function, one which assigns to each subset its choice set:

Proposition 2 *(Sen 1986) Given a relation $\leq$ over a finite set Ω, the choice set operation C defines a choice function iff $\leq$ is acyclic.*[3]

If a relation is not acyclic, elements involved in a cycle are said to be in a *conflict* because we cannot order them:

Definition 4 *Given a relation $<$ over a finite set Ω, x and y are in a* conflict wrt $<$ *iff there exist $w_0, \ldots, w_n, z_0, \ldots, z_m \in \Omega$ such that $x = w_0 < \cdots < w_n = y = z_0 < \cdots < z_m = x$, where $x, y \in \Omega$.*

3 Aggregation in Social Choice

We are interested in belief aggregation, but the community historically most interested in aggregation has been that of Social Choice theory. The aggregation is over preferences rather than beliefs, so the discussion in this subsection will focus on representing preferences; however, as we shall see, the results are equally relevant to representing beliefs. In the Social Choice community, the standard representation of an agent's preferences is a total pre-order. Each total pre-order $\preceq_i$ is interpreted as describing the weak preferences of an individual i, so that $x \preceq_i y$ means i considers alternative x to be at least as preferable as alternative y.[4] If $x \preceq_i y$ and $y \preceq_i x$, then i is indifferent between x and y.

Unfortunately, Arrow's Impossibility Theorem (Arrow 1963) showed that no aggregation operator over total pre-orders exists satisfying the following small set of desirable properties:

Definition 5 *Let f be an aggregation operator over the preferences $\preceq_1, \ldots, \preceq_n$ of n individuals, respectively, over a finite set of alternatives Ω, and let $\preceq = f(\preceq_1, \ldots, \preceq_n)$.*

- Restricted Range: *The range of f is the set of total pre-orders over Ω.*
- Unrestricted Domain: *The domain of f is the set of n-tuples of total pre-orders over Ω.*
- Pareto Principle: *If $x \prec_i y$ for all i, then $x \prec y$.*
- Independence of Irrelevant Alternatives (IIA): *Suppose $\preceq' = f(\preceq'_1, \ldots, \preceq'_n)$. If, for $x, y \in \Omega$, $x \preceq_i y$ iff $x \preceq'_i y$ for all i, then $x \preceq y$ iff $x \preceq' y$.*
- Non-Dictatorship: *There is no individual i such that, for every tuple in the domain of f and every $x, y \in \Omega$, $x \prec_i y$ implies $x \prec y$.*

Proposition 3 *(Arrow 1963) There is no aggregation operator that satisfies restricted range, unrestricted domain, (weak) Pareto principle, independence of irrelevant alternatives, and nondictatorship.*

This impossibility theorem led researchers to look for weakenings to Arrow's framework that would circumvent the result. One was to weaken the restricted range condition, requiring that the result of an aggregation only satisfy totality and quasi-transitivity rather than the full transitivity of a total pre-order. This weakening was sufficient to guarantee the existence of an aggregation function satisfying the other conditions, while still producing relations that defined choice functions (Sen 1986). However, this solution was not without its own problems.

First, total, quasi-transitive relations have unsatisfactory semantics. If $\preceq$ is total and quasi-transitive but not a total pre-order, its indifference relation is not transitive:

Proposition 4 *Let $\preceq$ be a relation over a finite set Ω and let $\sim$ be its symmetric restriction (i.e., $x \sim y$ iff $x \preceq y$ and $y \preceq x$). If $\preceq$ is total and quasi-transitive but not transitive, then $\sim$ is not transitive.*

There has been much discussion as to whether or not indifference should be transitive; in many cases one feels indifference should be transitive. If Deb enjoys plums and mangoes equally and also enjoys mangoes and peaches equally, we would conclude that she also enjoys plums and peaches equally. It seems that total quasi-transitive relations that are not total pre-orders cannot be understood easily as preference or indifference.

Since the existence of a choice function is generally sufficient for classical Social Choice problems, this issue was at least ignorable. However, in iterated aggregation, the result of the aggregation must not only be usable for making decisions, but must be interpretable as

[3] Sen's uses a slightly stronger definition of choice sets, but the theorem still holds in our more general case.

[4] The direction of the relation symbol is unintuitive, but standard practice in the belief revision community.

a new preference relation that may be involved in later aggregations; consequently, it must maintain clean semantics.

Secondly, the totality assumption is excessively restrictive for representing aggregate preferences. In general, a binary relation $\preceq$ can express four possible relationships between a pair of alternatives a and b: $a \preceq b$ and $b \not\preceq a$, $b \preceq a$ and $a \not\preceq b$, $a \preceq b$ and $b \preceq a$, and $a \not\preceq b$ and $b \not\preceq a$. Totality reduces this set to the first three which, under the interpretation of relations as representing weak preference, correspond to the two strict orderings of a and b, and indifference. However, consider the situation where a couple is trying to choose between an Italian and an Indian restaurant, but one strictly prefers Italian food to Indian food, whereas the second strictly prefers Indian to Italian. The couple's opinions are in conflict, a situation that does not fit into any of the three remaining categories. Thus, the totality assumption is essentially an assumption that conflicts do not exist. This, one may argue, is appropriate if we want to represent preferences of one agent (but see (Kahneman and Tversky 1979) for persuasive arguments that individuals *are* often ambivalent). However, the assumption is inappropriate if we want to represent aggregate preferences since individuals will almost certainly have differences of opinion.

4 Generalized Belief States

Let us turn to the domain of belief aggregation. A total pre-order over the set of possible worlds is a fairly well-accepted representation for a belief state in the belief revision community (Grove 1988; Katsuno and Mendelzon 1991; Lehmann and Magidor 1992; Gärdenfors and Makinson 1994). Instead of preference, relations represent relative likelihood, instead of indifference, equal likelihood. For the remainder of the paper, assume we are given some language $\mathcal{L}$ with a satisfaction relation $\models$ for $\mathcal{L}$. Let $\mathcal{W}$ be a finite, non-empty set of possible worlds (interpretations) over $\mathcal{L}$. Suppose $\preceq$ is a total pre-order on $\mathcal{W}$. The belief revision literature maintains that the conditional belief "if p then q" (where p and q are sentences in $\mathcal{L}$) holds if all the worlds in the choice set of those satisfying p also satisfy q; we write $Bel(p?q)$. The individual's unconditional beliefs are all those where p is the sentence *true*. If neither the belief $p?q$ nor its negation hold in the belief state, it is said to be *agnostic* with respect to $p?q$, written $Agn(p?q)$.

It should come as no surprise that belief aggregation is formally similar to preference aggregation and, as a result, is also susceptible to the problems described in the previous section. We propose a solution to these problems which generalizes the total pre-order representation so as to capture information about conflicts.

4.1 Modular, transitive states

We take strict likelihood as primitive. Since strict likelihood is not necessarily total, it is possible to represent agnosticism and conflicting opinions in the same structure. This choice deviates from that of most authors, but are similar to those of Kreps (Kreps 1990, p. 19) who is interested in representing both indifference and incomparability. Unlike Kreps, rather than use an asymmetric relation to represent strict likelihood (e.g., the strict version of a weak likelihood relation), we impose the less restrictive condition of modularity.

We formally define *generalized belief states*:

Definition 6 *A* generalized belief state $\prec$ *is a modular, transitive relation over* $\mathcal{W}$. *The set of possible generalized belief states over* $\mathcal{W}$ *is denoted* $\mathcal{B}$.

We interpret $a \prec b$ to mean "there is reason to consider a as strictly more likely than b." We represent equal likelihood, which we also refer to as "agnosticism," with the relationship $\sim$ defined such that $x \sim y$ if and only if $x \not\prec y$ and $y \not\prec x$. We define the conflict relation corresponding to $\prec$, denoted ∞, so that $x \infty y$ iff $x \prec y$ and $y \prec x$. It describes situations where there are reasons to consider either of a pair of worlds as strictly more likely than the other. In fact, one can easily check that ∞ precisely represents conflicts in a belief state in the sense of Definition 4.

For convenience, we will refer to generalized belief states simply as belief states for the remainder of the paper except when to do so would cause confusion.

4.2 Discussion

Let us consider why our choice of representation is justified. First, we agree with the Social Choice community that strict likelihood should be transitive.

As we discussed in the previous section, there is often no compelling reason why agnosticism/indifference should not be transitive; we also adopt this view. However, transitivity of strict likelihood by itself does not guarantee transitivity of agnosticism. A simple example is the following: $\prec = \{(a, c)\}$, so that $\sim = \{(a, b), (b, c)\}$. However, if we buy that strict likelihood should be transitive, then agnosticism is transitive identically when strict likelihood is also modular:

Proposition 5 *Suppose a relation* $\prec$ *is transitive and*

$\sim$ is the corresponding agnosticism relation. Then $\sim$ is transitive iff $\prec$ is modular.

In summary, transitivity and modularity are necessary if strict likelihood and agnosticism are both required to be transitive.

We should point out that conflicts are also transitive in our framework. At first glance, this may appear undesirable: it is entirely possible for a group to disagree on the relative likelihood of worlds a and b, and b and c, yet agree that a is more likely than c. However, we note that this transitivity follows from the cycle-based definition of conflicts (Definition 4), not from our belief state representation. It highlights the fact that we are not only concerned with conflicts that arise from simple disagreements over pairs of alternatives, but those that can be inferred from a series of inconsistent opinions as well.

Now, to argue that modular, transitive relations are sufficient to capture relative likelihood, agnosticism, and conflicts among a group of information sources, we first point out that adding irreflexivity would give us the class of relations that are strict versions of total pre-orders, i.e., conflict-free. Let $\mathcal{T}$ be the set of total pre-orders over $\mathcal{W}$, $\mathcal{T}_<$, the set of their strict versions.

Proposition 6 *The set of irreflexive relations in $\mathcal{B}$ is isomorphic to $\mathcal{T}$ and, in fact, equals $\mathcal{T}_<$.*

Secondly, the following representation theorem shows that each belief state partitions the possible worlds into sets of worlds either all equally likely or all potentially involved in a conflict, and totally orders these sets; worlds in distinct sets have the same relation to each other as do the sets.

Proposition 7 *$\prec \in \mathcal{B}$ iff there is a partition $\mathbf{W} = \langle W_0, \ldots, W_n \rangle$ of $\mathcal{W}$ such that:*

1. *For every $x \in W_i$ and $y \in W_j$, $i \neq j$ implies $i < j$ iff $x \prec y$.*
2. *Every W_i is either fully connected ($w \prec w'$ for all $w, w' \in W_i$) or fully disconnected ($w \not\prec w'$ for all $w, w' \in W_i$).*

Figure 1 shows three examples of belief states: one which is a total pre-order, one which is the strict version of a total pre-order, and one which is neither.

Thus, generalized belief states are not a big change from the strict versions of total pre-orders. They merely generalize these by weakening the assumption that sets of worlds not strictly ordered are equally likely, allowing for the possibility of conflicts. Now

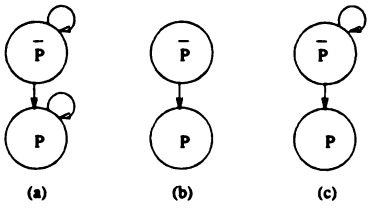


Figure 1: Three examples of generalized belief states: (a) a total pre-order, (b) the strict version of a total pre-order, (c) neither. (Each circle represents all the worlds in $\mathcal{W}$ which satisfy the sentence inside. An arc between circles indicates that $w \prec w'$ for every w in the head circle and w' in the tail circle; no arc indicates that $w \not\prec w'$ for each of these pairs. In particular, the set of worlds represented by a circle is fully connected if there is an arc from the circle to itself, fully disconnected otherwise.)

we can distinguish between agnostic and conflicting conditional beliefs. A belief state $\prec$ is agnostic about conditional belief $p?q$ (i.e., $Agn(p?q)$) if the choice set of worlds satisfying p contains both worlds which satisfy q and $\neg q$ and is fully disconnected. It is in conflict about this belief, written $Con(p?q)$, if the choice set is fully connected.

Finally, we compare the representational power of our definitions to those discussed in the previous section. First, $\mathcal{B}$ subsumes the class of total pre-orders:

Proposition 8 *$\mathcal{T} \subset \mathcal{B}$ and is the set of reflexive relations in $\mathcal{B}$.*

Secondly, $\mathcal{B}$ neither subsumes nor is subsumed by the set of total, quasi-transitive relations, and the intersection of the two classes is $\mathcal{T}$. Let $\mathcal{Q}$ be the set of total, quasi-transitive relations over $\mathcal{W}$, and $\mathcal{Q}_<$, the set of their strict versions.

Proposition 9

1. *$\mathcal{Q} \cap \mathcal{B} = \mathcal{T}$.*
2. *$\mathcal{B} \not\subseteq \mathcal{Q}$.*
3. *$\mathcal{Q} \not\subseteq \mathcal{B}$ if $\mathcal{W}$ has at least three elements.*
4. *$\mathcal{Q} \subset \mathcal{B}$ if $\mathcal{W}$ has one or two elements.*

Because modular, transitive relations represent strict preferences, it is probably fairer to compare them to the class of strict versions of total, quasi-transitive relations. Again, neither class subsumes the other, but this time the intersection is $\mathcal{T}_<$:

Proposition 10

1. $\mathcal{Q}_< \cap \mathcal{B} = \mathcal{T}_<$.
2. $\mathcal{B} \not\subseteq \mathcal{Q}_<$.
3. $\mathcal{Q}_< \not\subseteq \mathcal{B}$ *if* $\mathcal{W}$ *has at least three elements.*
4. $\mathcal{Q}_< \subset \mathcal{B}$ *if* $\mathcal{W}$ *has one or two elements.*

In the next section, we define a natural aggregation policy based on this new representation that admits clear semantics and obeys appropriately modified versions of Arrow's conditions.

5 Single-agent belief state construction

Suppose an agent is informed by a set of sources, each with its individual belief state. Suppose further that the agent has ranked the sources by level of credibility. We propose an operator for constructing the agent's belief state $\prec$ by aggregating the belief states of the sources in S while accounting for the credibility ranking of the sources.

Example 1 *We will use a running example from our space robot domain to help provide intuition for our definitions. The robot sends to earth a stream of telemetry data gathered by the spacecraft, as long as it receives positive feedback that the data is being received. At some point it loses contact with the automatic feedback system, so it sends a request for information to an agent on earth to find out if the failure was caused by a failure of the feedback system or by an overload of the data retrieval system. In the former case, it would continue to send data, in the latter, desist. As it so happens, there has been no overload, but the computer running the feedback system has hung. The agent consults the following three experts, aggregates their beliefs, and sends the results back to the robot:*

1. s_p, *the computer programmer that developed the feedback program, believes nothing could ever go wrong with her code, so there must have been an overload problem. However, she admits that if her program had crashed, the problem could ripple through to cause an overload.*
2. s_m, *the manager for the telemetry division, unfortunately has out-dated information that the feedback system is working. She was also told by the engineer who sold her the system that overloading could never happen. She has no idea what would happen if there was an overload or the feedback system crashed.*
3. s_t, *the technician working on the feedback system, knows that the feedback system crashed, but doesn't know whether there was a data-overload. Not being familiar with the retrieval system, she is also unable to speculate whether the data retrieval system would have overloaded if the feedback system had not failed.*

Let F and D be propositional variables representing that the feedback and data retrieval systems, respectively, are okay. The belief states for the three sources are shown in Figure 2.

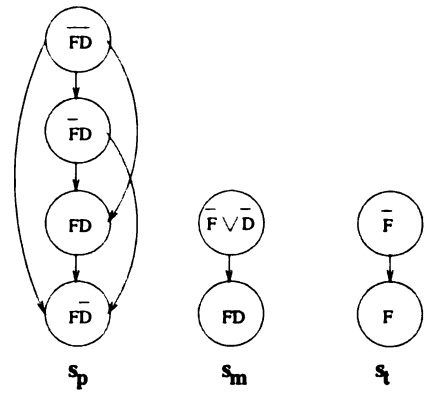


Figure 2: The belief states of s_p, s_m, and s_t in Example 1.

Let us begin the formal development by defining sources:

Definition 7 $\mathcal{S}$ *is a finite set of* sources. *With each source $s \in \mathcal{S}$ is associated a belief state $<^s \in \mathcal{B}$.*

We denote the agnosticism and conflict relations of a source s by $\approx^s$ and $\bowtie^s$, respectively. It is possible to assume that the belief state of a source is conflict free, i.e., acyclic. However, this is not necessary if we allow sources to suffer from the human malady of "being torn between possibilities."

We assume that the agent's credibility ranking over the sources is a total pre-order:

Definition 8 $\mathcal{R}$ *is a totally ordered finite set of* ranks.

Definition 9 *rank* $: \mathcal{S} \to \mathcal{R}$ *assigns to each source a rank.*

Definition 10 $\sqsupseteq$ *is the total pre-order over $\mathcal{S}$ induced by the ordering over $\mathcal{R}$. That is, $s \sqsupseteq s'$ iff*

$rank(s) \geq rank(s')$; we say s' is as credible as *s. $\sqsupseteq_S$ is the restriction of $\sqsupseteq$ to $S \subseteq \mathcal{S}$.*

We use $\sqsupset$ and $\equiv$ to denote the asymmetric and symmetric restrictions of $\sqsupseteq$, respectively.[5] The finiteness of $\mathcal{S}$ ($\mathcal{R}$) ensures that a maximal source (rank) always exists, which is necessary for some of our results. Weaker assumptions are possible, but at the price of unnecessarily complicating the discussion.

We are ready to consider the source aggregation problem. In the following, assume an agent is informed by a set of sources $S \subseteq \mathcal{S}$. We look at two special cases—equal-ranked and strictly-ranked source aggregation—before considering the general case.

5.1 Equal-ranked sources aggregation

Suppose all the sources have the same rank so that $\sqsupseteq_S$ is fully connected. Intuitively, we want take all offered opinions seriously, so we take the union of the relations:

Definition 11 *If $S \subseteq \mathcal{S}$, then $Un(S)$ is the relation $\bigcup_{s \in S} <^s$.*

By simply taking the union of the source belief states, we may lose transitivity. However, we do not lose modularity:

Proposition 11 *If $S \subseteq \mathcal{S}$, then $Un(S)$ is modular but not necessarily transitive.*

Thus, we know from Proposition 1 that we need only take the transitive closure of $Un(S)$ to get a belief state:

Definition 12 *If $S \subseteq \mathcal{S}$, then $AGRUn(S)$ is the relation $Un(S)^+$.*

Proposition 12 *If $S \subseteq \mathcal{S}$, then $AGRUn(S) \in \mathcal{B}$.*

Not surprisingly, by taking all opinions of all sources seriously, we may generate many conflicts, manifested as fully connected subsets of $\mathcal{W}$.

Example 2 *Suppose all three sources in the space robot scenario of Example 1 are considered equally credible, then the aggregate belief state will be the fully connected relation indicating that there are conflicts over every belief.*

[5] Note that, unlike the relations representing belief states, $\geq$ and $\sqsupseteq$ are read in the intuitive way, that is, "greater" corresponds to "better."

5.2 Strictly-ranked sources aggregation

Next, consider the case where the sources are strictly ranked, i.e., $\sqsupseteq_S$ is a total order. We define an operator such that lower-ranked sources refine the belief states of higher ranked sources. That is, in determining the ordering of a pair of worlds, the opinions of higher-ranked sources generally override those of lower-ranked sources, and lower-ranked sources are consulted when higher-ranked sources are agnostic:

Definition 13 *If $S \subseteq \mathcal{S}$, then $AGRRf(S)$ is the relation*
$$\left\{(x, y) : \exists s \in S.\ x <^s y \wedge \left(\forall s' \sqsupset s \in S.\ x \approx^{s'} y\right)\right\}.$$

The definition of the $AGRRf$ operator does not rely on $\sqsupseteq_S$ being a total order, and we will use it in this more general setting in the following sub-section. However, in the case that $\sqsupseteq_S$ is a total order, the result of applying $AGRRf$ is guaranteed to be a belief state.

Proposition 13 *If $S \subseteq \mathcal{S}$ and $\sqsupseteq_S$ is a total order, then $AGRRf(S) \in \mathcal{B}$.*

Example 3 *Suppose, in the space robot scenario of Example 1, the technician is considered more credible than the manager who, in turn, is considered more credible than the programmer. The aggregate belief state, shown in Figure 3, informs the robot correctly that the feedback system has crashed, but that it shouldn't worry about an overload problem and should keep sending data.*

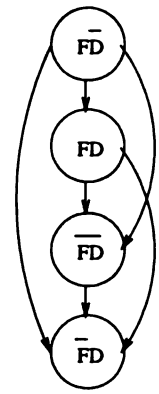


Figure 3: The belief state after aggregation in Example 3 when $s_t \sqsupset s_m \sqsupset s_p$.

Note that this case of strictly-ranked sources is almost exactly that considered in (Maynard-Reid II and Shoham 2000), except that the authors are not able to allow for conflicts in belief states. A surprising result they show is that standard AGM belief revision (Alchourrón *et al.* 1985) can be modeled as the aggrega-

tion of two sources, the informant and the informee, where the informant is considered more credible than the informee.

5.3 General aggregation

In the general case, we may have several ranks represented and multiple sources of each rank. It will be instructive to first consider the following seemingly natural strawman operator, AGR^*: First combine equi-rank sources using $AGRUn$, then aggregate the strictly-ranked results using what is essentially $AGRRf$:

Definition 14 *Let $S \subseteq \mathcal{S}$. For any $r \in \mathcal{R}$, let $<_r = AGRUn(\{s \in S : rank(s) = r\})$ and $\approx_{r'}$, the corresponding agnosticism relation. Also, let $rank(S) = \{r \in \mathcal{R} : \exists s \in S.\ rank(s) = r\}$. $AGR^*(S)$ is the relation*

$$\left\{(x,y) : \begin{array}{l} \exists r \in \mathcal{R}.\ x <_r y \wedge \\ (\forall r' > r \in ranks(S).\ x \approx_{r'} y) \end{array}\right\}$$

AGR^* indeed defines a legitimate belief state:

Proposition 14 *If $S \subseteq \mathcal{S}$, then $AGR^*(S) \in \mathcal{B}$.*

Unfortunately, a problem with this "divide-and-conquer" approach is it assumes the result of aggregation is independent of potential interactions between the individual sources of different ranks. Consequently, opinions that will eventually get overridden may still have an indirect effect on the final aggregation result by introducing superfluous opinions during the intermediate equi-rank aggregation step, as the following example shows:

Example 4 *Let $\mathcal{W} = \{a, b, c\}$. Suppose $S \subseteq \mathcal{S}$ such that $S = \{s_0, s_1, s_2\}$ with belief states $<^{s_0} = \{(b,a),(b,c)\}$ and $<^{s_1} = <^{s_2} = \{(a,b),(c,b)\}$, and where $s_2 \sqsupset s_1 \equiv s_0$. Then $AGR^*(S)$ is $\{(a,b),(c,b),(a,c),(c,a),(a,a),(b,b),(c,c)\}$. All sources are agnostic over a and c, yet (a,c) and (c,a) are in the result because of the transitive closure in the lower rank involving opinions ((b,c) and (b,a)) which actually get overridden in the final result.*

Because of these undesired effects, we propose another aggregation operator which circumvents this problem by applying refinement (as defined in Definition 13) to the set of source belief states before infering new opinions via closure:

Definition 15 *The* rank-based aggregation *of a set of sources $S \subseteq \mathcal{S}$ is $AGR(S) = AGRRf(S)^+$.*

Encouragingly, AGR outputs a valid belief state:

Proposition 15 *If $S \subseteq \mathcal{S}$, then $AGR(S) \in \mathcal{B}$.*

Example 5 *Suppose, in the space robot scenario of Example 1, the technician is still considered more credible than the manager and the programmer, but the latter two are considered equally credible. The aggregate belief state, shown in Figure 5, still gives the robot the correct information about the state of the system. The robot also learns for future reference that there is some disagreement over whether or not there would have been a data overload if the feedback system were working.*

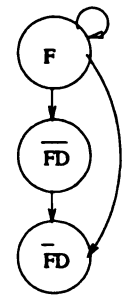


Figure 4: The belief state after aggregation in Example 5 when $s_t \sqsupset s_m \equiv s_p$.

We observe that AGR, when applied to the set of sources in Example 4, does indeed bypass the problem described above of extraneous opinion introduction:

Example 6 *Assume $\mathcal{W}$, S, and $\sqsupseteq$ are as in Example 4. $AGR(S) = \{(a,b),(c,b)\}$.*

We also observe that AGR behaves well in the special cases we've considered, reducing to $AGRUn$ when all sources have equal rank, and to $AGRRf$ when the sources are totally ranked:

Proposition 16 *Suppose $S \subseteq \mathcal{S}$.*

1. *If $\sqsupseteq_S$ is fully connected, $AGR(S) = AGRUn(S)$.*
2. *If $\sqsupseteq_S$ is a total order, $AGR(S) = AGRRf(S)$.*

5.4 Arrow, revisited

Finally, a strong argument in favor of AGR is that it satisfies appropriate modifications of Arrow's conditions. Let f be an operator which aggregates the belief states $<^{s_1}, \ldots, <^{s_n}$ over $\mathcal{W}$ of n sources $s_1, \ldots, s_n \in \mathcal{S}$, respectively, and let $\prec = f(<^{s_1}, \ldots, <^{s_n})$. We consider each condition separately.

Restricted range The output of the aggregation function will be a modular, transitive belief state rather than a total pre-order.

Definition 16 (modified) Restricted Range: *The range of f is $\mathcal{B}$.*

Unrestricted domain Similarly, the input to the aggregation function will be modular, transitive belief states of sources rather than total pre-orders.

Definition 17 (modified) Unrestricted Domain: *For each i, $<^{s_i}$ can be any member of $\mathcal{B}$.*

Pareto principle Generalized belief states already represent strict likelihood. Consequently, we use the actual input and output relations of the aggregation function in place of their strict versions to define the Pareto principle. Obviously, because we allow for the introduction of conflicts, AGR will not satisfy the original formal Pareto principle which essentially states that if all sources have an unconflicted belief that one world is strictly more likely than another, this must also be true of the aggregated belief state. Neither condition is necessarily stronger than the other.

Definition 18 (modified) Pareto Principle: *If $x <^{s_i} y$ for all i, then $x \prec y$.*

Independence of irrelevant alternatives Conflicts are defined in terms of cycles, not necessarily binary. By allowing the existence of conflicts, we effectively have made it possible for outside worlds to affect the relation between a pair of worlds, viz., by involving them in a cycle. As a result, we need to weaken IIA to say that the relation between worlds should be independent of other worlds **unless** these other worlds put them in conflict.

Definition 19 (modified) Independence of Irrelevant Alternatives (IIA): *Suppose $s'_1, \ldots, s'_n \in \mathcal{S}$ such that $s_i \equiv s'_i$ for all i, and $\prec' = f(<^{s'_1}, \ldots, <^{s'_n})$. If, for $x, y \in \mathcal{W}$, $x <^{s_i} y$ iff $x <^{s'_i} y$ for all i, $x \not\diamond y$, and $x \not\diamond' y$, then $x \prec y$ iff $x \prec' y$.*

Non-dictatorship As with the Pareto principle definition, we use the actual input and output relations to define non-dictatorship since belief states represent strict likelihood. From this perspective, our setting requires that informant sources of the highest rank be "dictators" in the sense considered by Arrow. However, the setting originally considered by Arrow was one where all individuals are ranked equally. Thus, we make this explicit in our new definition of non-dictatorship by adding the pre-condition that all sources be of equal rank. Now, AGR treats a set of equi-rank sources equally by taking all their opinions seriously, at the price of introducing conflicts. So, intuitively, there are no dictators. However, because Arrow did not account for conflicts in his formulation, all the sources will be "dictators" by his definition. We need to modify the definition of non-dictatorship to say that no source can always push opinions through without them ever being contested.

Definition 20 (modified) Non-Dictatorship: *If $s_i \equiv s_j$ for all i, j, then there is no i such that, for every combination of source belief states and every $x, y \in \mathcal{W}$, $x <^{s_i} y$ and $y \not<^{s_i} x$ implies $x \prec y$ and $y \not\prec x$.*

We now show that AGR indeed satisfies these conditions:

Proposition 17 *Let $S = \{s_1, \ldots, s_n\} \subseteq \mathcal{S}$ and $AGR_f(<^{s_1}, \ldots, <^{s_n}) = AGR(S)$. AGR_f satisfies (the modified versions of) restricted range, unrestricted domain, Pareto principle, IIA, and non-dictatorship.*

6 Multi-agent fusion

So far, we have only considered the case where a single agent must construct or update her belief state once informed by a set of sources. Multi-agent fusion is the process of aggregating the belief states of a set of agents, each with its respective set of informant sources. We proceed to formalize this setting.

An agent A is *informed by* a set of sources $S \subseteq \mathcal{S}$. Agent A's *induced belief state* is the belief state formed by aggregating the belief states of its informant sources, i.e., $AGR(S)$. Assume the set of agents to fuse agree upon *rank* (and, consequently, $\sqsupseteq$).[6] We define the fusion of this set to be an agent informed by the combination of informant sources:

Definition 21 *Let $\mathcal{A} = \{A_1, \ldots, A_n\}$ be a set of agents such that each agent A_i is informed by $S_i \subseteq \mathcal{S}$. The* fusion *of $\mathcal{A}$, written $\bigtriangledown(\mathcal{A})$, is an agent informed by $S = \bigcup_{i=1}^{n} S_i$.*

[6] We could easily extend the framework to allow for individual rankings, but we felt that the small gain in generality would not justify the additional complexity and loss of perspicuity. Similarly, we could consider each agent as having a credibility ordering only over its informant sources. However, it is unclear how, for example, credibility orderings over disjoint sets of sources should be combined into a new credibility ordering since their union will not be total.

Not surprisingly given its set-theoretic definition, fusion is idempotent, commutative, and associative. These properties guarantee the invariance required in multi-agent belief aggregation applications such as our space robot domain.

In the multi-agent space robot scenario described in Section 1, we only have a direct need for the belief states that result from fusion. We are only interested in the belief states of the original sources in as far as we want the fused belief state to reflect its informant history. An obvious question is whether it is possible to compute the belief state induced by the agents' fusion solely from their initial belief states, that is, without having to reference the belief states of their informant sources. This is highly desirable because of the expense of storing—or, as in the case of our space robot example, transmitting—all source belief states; we would like to represent each agent's knowledge as compactly as possible.

In fact, we can do this if all sources have equal rank. We simply take the transitive closure of the union of the agents' belief states:

Proposition 18 *Let $\mathcal{A}$ and S be as in Definition 21, $\prec^{A_i}$, agent A_i's induced belief state, and $\sqsupseteq_S$, fully connected. If $A = \bigcirc\!\!\!\!\vee(\mathcal{A})$, then $\left(\bigcup_{A_i \in \mathcal{A}} \prec^{A_i}\right)^+$ is A's induced belief state.*

Unfortunately, the equal rank case is special. If we have sources of different ranks, we generally cannot compute the induced belief state after fusion using only the agent belief states before fusion, as the following simple example demonstrates:

Example 7 *Let $\mathcal{W} = \{a, b\}$. Suppose two agents A_1 and A_2 are informed by sources s_1 with belief state $<^{s_1} = \{(a, b)\}$ and s_2 with belief state $<^{s_2} = \{(b, a)\}$, respectively. A_1's belief state is the same as s_1's and A_2's is the same as s_2's. If $s_1 \sqsupset s_2$, then the belief state induced by $\bigcirc\!\!\!\!\vee(A_1, A_2)$ is $<^{s_1}$, whereas if $s_2 \sqsupset s_1$, then it is $<^{s_2}$. Thus, just knowing the belief states of the fused agents is not sufficient for computing the induced belief state. We need more information about the original sources.*

However, if sources are totally pre-ordered by credibility, we can still do much better than storing all the original sources. It is enough to store for each opinion of $AGRRf(S)$ the rank of the highest-ranked source supporting it. We define *pedigreed belief states* which enrich belief states with this additional information:

Definition 22 *Let A be an agent informed by a set of sources $S \subseteq \mathcal{S}$. A's* pedigreed belief state *is a pair $(\prec, l)$ where $\prec = AGRRf(S)$ and $l : \prec \rightarrow \mathcal{R}$ such that $l((x, y)) = \max\{rank(s) : x <^s y, s \in S\}$. We use $\prec_r^A$ to denote the* restriction of A's pedigreed belief state to r, *that is, $\prec_r^A = \{(x, y) \in \prec : l((x, y)) = r\}$.*

We verify that a pair's label is, in fact, the rank of the source used to determine the pair's membership in $AGRRf(S)$, not that of some higher-ranked source:

Proposition 19 *Let A be an agent informed by a set of sources $S \subseteq \mathcal{S}$ and with pedigreed belief state $(\prec, l)$. Then*

$$x \prec_r^A y$$

iff

$$\exists s \in S.\ x <^s y \wedge r = rank(s) \wedge \left(\forall s' \sqsupset s \in S.\ x \approx^{s'} y\right).$$

The belief state induced by a pedigreed belief state $(\prec, l)$ is, obviously, the transitive closure of $\prec$.

Now, given only the pedigreed belief states of a set of agents, we can compute the new pedigreed belief state after fusion. We simply combine the labeled opinions using our refinement techniques.

Proposition 20 *Let $\mathcal{A}$ and S be as in Definition 21, $\sqsupseteq_S$, a total pre-order, and $A = \bigcirc\!\!\!\!\vee(\mathcal{A})$. If*

1. *$\prec$ is the relation*

$$\left\{(x, y) : \begin{array}{c} \exists A_i \in \mathcal{A}, r \in \mathcal{R}.\ x \prec_r^{A_i} y \wedge \\ \left(\forall A_j \in \mathcal{A}, r' > r \in \mathcal{R}.\ x \sim_{r'}^{A_j} y\right) \end{array}\right\}$$

   over $\mathcal{W}$,

2. *$l : \prec \rightarrow \mathcal{R}$ such that $l((x, y)) = \max\{r : x \prec_r^{A_i} y, A_i \in \mathcal{A}\}$, and*

then $(\prec, l)$ is A's pedigreed belief state.

From the perspective of the induced belief states, we are essentially discarding unlabeled opinions (i.e., those derived by the closure operation) before fusion. Intuitively, we are learning new information so we may need to retract some of our inferred opinions. After fusion, we re-apply closure to complete the new belief state. Interestingly, in the special case where the sources are strictly-ranked, the closure is unnecessary:

Proposition 21 *If $\mathcal{A}$ and S are as in Definition 21, $\sqsupseteq_S$ is a total order, and $(\prec, l)$ is the pedigreed belief state of $\bigcirc\!\!\!\!\vee(\mathcal{A})$, then $\prec^+ = \prec$.*

Example 8 *Let's look once more at the space robot scenario considered in Example 1. Suppose the arrogant programmer is not part of the telemetry team, but instead works for a company on the other side of the country. Then the robot has to request information from two separate agents, one to query the manager and technician and one to query the programmer. Assume that the agents and the robot all rank the sources the same, assigning the technician rank 2 and the other two agents rank 1, which induces the same credibility ordering used in Example 5. The agents' pedigreed belief states and the result of their fusion are shown in Figure 5.*

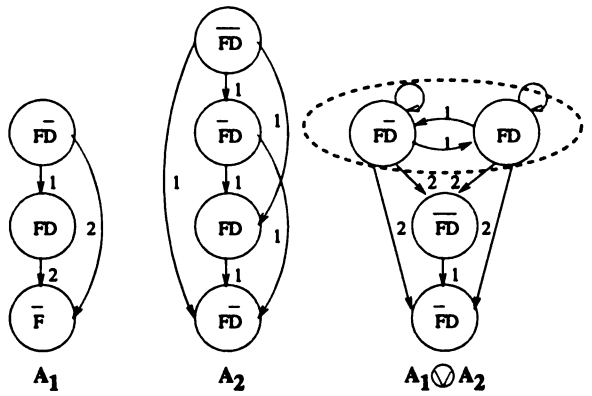


Figure 5: The pedigreed belief states of agent A_1 informed by s_m and s_t and of agent A_2 informed by s_p, and the result of their fusion in Example 8.

The first agent does not provide any information about overloading and the second agent provides incorrect information. However, we see that after fusing the two, the robot has a belief state that is identical to what it computed in Example 5 when there was only one agent informed by all three sources (we've only separated the top set of worlds so as to show the labeling). Consequently, it now knows the correct state of the system. And, satisfyingly, the final result does not depend on the order in which the robot receives the agents' reports.

The savings obtained in required storage space by this scheme can be substantial. Whereas explicitly storing all of an agent's informant sources S requires $\mathcal{O}(\|S\|2^{\mathcal{W}})$ amount of space in the worst case (when all the sources' belief states are fully connected relations), storing a pedigreed belief state only requires $\mathcal{O}(2^{\mathcal{W}})$ space in the worst case. Moreover, not only does the enriched representation allow us to conserve space, but it also provides for potential savings in the efficiency of computing fusion since, for each pair of worlds, we only need to consider the opinions of the *agents* rather than those of all the sources in the combined set of informants.

Incidentally, if we had used AGR^* as the basis for our general aggregation, simply storing the rank of the maximum supporting sources would not give us sufficient information to compute the induced belief state after fusion. To demonstrate this, we give an example where two pairs of sources induce the same annotated agent belief states, yet yield different belief states after fusion:

Example 9 *Let $\mathcal{W}$, $\mathcal{S}$, and $\sqsupseteq$ be as in Example 4. Suppose agents A_1, A_2, A_1', and A_2' are informed by sets of sources S_1, S_2, S_1', and S_2', respectively, where $S_1 = S_2 = \{s_2\}$, $S_1' = \{s_0, s_2\}$, and $S_2' = \{s_1, s_2\}$. AGR^* dictates that the pedigreed belief states of all four agents equal $<^{s_2}$ with all opinions annotated with rank(s_2). In spite of this indistinguishability, if $A = \oslash(\{A_1, A_2\})$ and $A' = \oslash(\{A_1', A_2'\})$, then A's induced belief state equals $<^{s_2}$, i.e., $\{(a,b),(c,b)\}$, whereas A''s is $\{(a,b),(c,b),(a,c),(c,a),(a,a),(b,b),(c,c)\}$.*

7 Conclusion

We have described a semantically clean representation for aggregate beliefs which allows us to represent conflicting opinions without sacrificing the ability to make decisions. We have proposed an intuitive operator which takes advantage of this representation so that an agent can combine the belief states of a set of informant sources totally pre-ordered by credibility. Finally, we have described a mechanism for fusing the belief states of different agents which iterates well.

The aggregation methods we have discussed here are just special cases of a more general framework based on voting. That is, we account not only for the ranking of the sources supporting or disagreeing with an opinion (i.e., the *quality* of support), but also the percentage of sources in each camp (the *quantity* of support). Such an extension allows for a much more refined approach to aggregation, one much closer to what humans often use in practice. Exploring this richer space is the subject of further research.

Another problem which deserves further study is developing a fuller understanding of the properties of the *Bel*, *Agn*, and *Con* operators and how they interrelate.

Acknowledgements

Pedrito Maynard-Reid II was partly supported by a National Physical Science Consortium Fellowship. The final version of this paper was written with the

financial support of the Jean et Hélène Alfassa Fund for Research in Artificial Intelligence.

References

Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

Kenneth J. Arrow. *Social Choice and Individual Values*. Wiley, New York, 2nd edition, 1963.

Peter Gärdenfors and David Makinson. Nonmonotonic inference based on expectations. *Artificial Intelligence*, 65(1):197–245, January 1994.

Adam Grove. Two modellings for theory change. *Journal of Philosophical Logic*, 17:157–170, 1988.

D. Kahneman and A. Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2):263–291, March 1979.

Hirofumi Katsuno and Alberto O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52(3):263–294, 1991.

David M. Kreps. *A Course in Microeconomic Theory*. Princeton University Press, 1990.

Daniel Lehmann and Menachem Magidor. What does a conditional knowledge base entail? *Artificial Intelligence*, 55(1):1–60, May 1992.

Pedrito Maynard-Reid II and Yoav Shoham. Belief fusion: Aggregating pedigreed belief states. *Journal of Logic, Language, and Information*, 2000. To appear.

Amartya Sen. Social choice theory. In K. J. Arrow and M. D. Intriligator, editors, *Handbook of Mathematical Economics*, volume III, chapter 22, pages 1073–1181. Elevier Science Publishers, 1986.

Automated Reasoning I

On Strongest Necessary and Weakest Sufficient Conditions

Fangzhen Lin (flin@cs.ust.hk)
Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Abstract

Given a propositional theory T and a proposition q, a sufficient condition of q is one that will make q true under T, and a necessary condition of q is one that has to be true for q to be true under T. In this paper, we propose a notion of strongest necessary and weakest sufficient conditions. Intuitively, the strongest necessary condition of a proposition is the most general consequence that we can deduce from the proposition under the given theory, and the weakest sufficient condition is the most general abduction that we can make from the proposition under the given theory. We show that these two conditions are dual ones, and can be naturally extended to arbitrary formulas. We investigate some computational properties of these two conditions and discuss some of their potential applications.

1 Introduction

Given a propositional theory T and a proposition q, a sufficient condition α of q is one that will make q true under T: $T \models \alpha \supset q$. Symmetrically, a necessary condition α of q is one that has to be true for q to be true under T: $T \models q \supset \alpha$.

For instance, consider the following theory T:

$$rain \supset grassWet,$$
$$sprinklerOn \supset grassWet.$$

A sufficient condition for $grassWet$ to be true under T is for $rain$ to be true. Another (trivial) one is $grassWet$ itself. Yet another one is $sprinklerOn$. And yet another one is $rain \vee sprinklerOn$.

In this paper we shall propose a notion of weakest sufficient and strongest necessary conditions, study their properties, and consider ways of computing them. There are many potential applications. The following are two examples:

- **Abduction** As we can see from the above example, given an observation, there may be more than one abductive conclusion that we can draw. It should be useful to find the weakest of such conclusions, i.e. the weakest sufficient condition of the observation.

- **Definability** It is often necessary to determine whether a given theory yields a definition of a proposition in terms of a set of other propositions. For example, such computation is essential in both Simon's [14] and Pearl's [12] approaches to causation. This is also what is needed in order to compute successor state axioms [13] from causal theories. As we shall see, the problem of whether a proposition can be defined in terms of other propositions, and if so, finding an explicit definition of this proposition can be reduced to that of computing the two conditions that we shall propose in this paper. Furthermore, we believe definability is best handled using these two conditions. While a proposition may or may not be definable in terms of a set of base propositions, our strongest necessary and weakest sufficient conditions, as we shall show, always exist and are unique up to logical equivalence under the given theory. Furthermore, these two conditions are useful to have even when a proposition cannot be defined in terms of others. For instance, in the context of reasoning about actions, this situation arises when an action has an indeterminate effect on a proposition. In this case, the strongest necessary and weakest sufficient conditions of this proposition can be used to completely capture the

effect of this action on the proposition.

This paper is organized as follows. We include logical preliminaries in section 2. In section 3, we define the notion of strongest necessary and weakest sufficient conditions, and prove some results that characterize these conditions. In section 4, we extend these two conditions to that of arbitrary formulas. In Section 5 we outline some algorithms for computing these two conditions and discuss some experimental results. Finally, we make some concluding remarks in section 6.

2 Logical preliminaries

We assume a propositional language. As usual, a truth assignment is a function from the set of propositions in the language to $\{true, false\}$, and the satisfiability relation $M \models \varphi$ is defined as usual between a truth assignment M and a formula φ.

Given a set P of propositions, a formula that mentions only propositions in P is called a *formula of* P. It is clear that the truth value of such a formula is determined by the truth values of the propositions in P only.

Given a formula φ, and a proposition p, we denote by $\varphi(p/true)$ the result of replacing every p in φ by $true$. We extend this notation to a set of formulas as well: if T is a finite set of formulas, then by $T(p/true)$ we mean the formula

$$\bigwedge_{\varphi \in T} \varphi(p/true).$$

We define $\varphi(p/false)$ and $T(p/false)$ similarly. It is clear that for any truth assignment M, $M \models \varphi(p/true)$ iff $M' \models \varphi$, where M' is the truth assignment that is exactly like M but $M'(p) = true$. A similar result holds for $\varphi(p/false)$.

A notion of forgetting, or eliminations of "middle terms" as Boole ([2], page 99) called it, will be essential here. Given a formula φ, and a proposition p, the result of forgetting p in φ, written $forget(\varphi; p)$, is the following formula:

$$\varphi(p/true) \vee \varphi(p/false).$$

Now given a finite set of propositions P, the result of forgetting P in φ, written $forget(\varphi; P)$, is defined inductively as:

$$forget(\varphi; \emptyset) = \varphi,$$
$$forget(\varphi; P \cup \{q\}) = forget(forget(\varphi; q); P).$$

It can be shown that for any formula φ and any propositions p_1 and p_2, $forget(forget(\varphi; p_1); p_2)$ and $forget(forget(\varphi; p_2); p_1)$ are equivalent. So $forget(\varphi; P)$ is well defined.

Since Boole [2] first defined it in 1854, this notion of forgetting, or eliminations of middle terms, has been re-discovered several times. Weber [16] re-introduced it and used it for updating propositional knowledge bases. It is also a special case of a more general notion of forgetting defined on first-order languages by Lin and Reiter [10] for capturing database progression [9] and certain notions of relevance in problem solving. Lewis [6] (page 155) observed that "For purposes of application of the algebra[1] to ordinary reasoning, elimination is a process more important than solution, since most processes of reasoning take place through the elimination of 'middle' terms." As we shall see, this notion of "the elimination of middle terms" is certainly central for us here.

The following lemma is entailed by Proposition 3 in (Lin and Reiter [10]), and will play an important role in this paper.

Lemma 1 *For any formula φ and any set P of propositions, if ϕ does not mention any propositions in P, then*

$$\varphi \models \phi \Leftrightarrow forget(\varphi; P) \models \phi.$$

In particular, we have that $\varphi \models forget(\varphi; P)$.

3 Strongest necessary and weakest sufficient conditions

We begin with strongest necessary conditions. As we shall see, weakest sufficient conditions are dual ones.

Definition 1 *Let T be a theory, P a set of propositions in T, and q a proposition in T but not in P. A formula φ of P is said to be a* necessary condition *of q on P under T if $T \models q \supset \varphi$. It is said to be a* strongest necessary condition *if it is a necessary condition, and for any other necessary condition φ', we have that $T \models \varphi \supset \varphi'$.*

In this paper, both the background theory T and the base set P of propositions are assumed to be finite. The following proposition is straightforward:

Proposition 1 *If both φ and φ' are strongest necessary conditions of q on P under T, then $T \models \varphi \equiv \varphi'$.*

[1] The author's note: "the algebra" = Boolean algebra as we know it today.

This means that if there is a strongest necessary condition, then it is unique up to logical equivalence under the background theory. So sometimes, we also call a strongest necessary condition *the* strongest one.

Example 1 The following examples provide some intuitions about our notion of strongest necessary conditions.

1. $T_1 = \{q \supset (p_1 \wedge p_2)\}$. The strongest necessary condition of q on $\{p_1, p_2\}$ under T_1 is $p_1 \wedge p_2$, and the strongest necessary condition of q on $\{p_1\}$ is p_1.
2. $T_2 = \{q \supset (p_1 \vee p_2)\}$. The strongest necessary condition of q on $\{p_1, p_2\}$ is $p_1 \vee p_2$, and the strongest necessary condition of q on $\{p_1\}$ is *true* because neither p_1 nor $\neg p_1$ follow from q under T_2.
3. $T_3 = \{q \supset p_1, q\}$. A strongest necessary condition of q on $\{p_1\}$ is p_1. Another one is *true*. Of course, $T_3 \models p_1 \equiv true$.

The following theorem captures this notion of strongest necessary conditions in terms of truth assignments. It says that a necessary condition φ of q is a strongest one on P if and only if for any model of T, if it satisfies φ, then it can be modified into a model of q without changing the truth values of the propositions in P:

Theorem 1 *Let T, P, and q be as in Definition 1. Let φ be a necessary condition of q on P under T. Then φ is a strongest necessary condition of q on P under T iff for any model M of T and φ, there is an assignment M' such that:*

1. *for any $p \in P$, $M'(p) = M(p)$.*
2. *$M' \models T \cup \{q\}$.*

Proof: The "if" part: suppose φ is the strongest necessary condition (SNC) of q, and $M \models T \cup \{\varphi\}$. We need to show the existence of a model M' that satisfies the two conditions in the theorem. We prove by contradiction. Suppose there is no such M', i.e. there is no model of $T \cup \{q, M(P)\}$, where $M(P)$ is the conjunction of p, if $M(p) = true$, or $\neg p$, if $M(p) = false$, for every $p \in P$. Then $T \models q \supset \neg M(P)$. This means that $\neg M(P)$ is a necessary condition (NC) of q on P. Thus $T \models \varphi \supset \neg M(P)$, this contradicts with the assumption that $M \models T \cup \{\varphi\}$.

The "only if" part: suppose for any model M of T and φ, there is another model M' that satisfies the two conditions in the theorem. We need to show that φ is the SNC of q. Let φ' be another NC of q: it is a formula of P and $T \models q \supset \varphi'$. We need to prove that $T \models \varphi \supset \varphi'$. Let M be any model of φ and T. We show that M is also a model of φ'. By our assumption, there is another model M' that satisfies the two conditions in the theorem. This means that $M' \models q$ and $M' \models T$. So $M' \models \varphi'$. But $M(P) = M'(P)$, and φ' is a formula of P. So we have $M \models \varphi'$ as well. ■

Notice that it is essential that the formula φ in Theorem 1 be a necessary condition of q is necessary. Otherwise, we could always let φ be *false*.

We can use this theorem to verify that a certain necessary condition is in fact the strongest. For instance, consider $T_1 = \{q \supset p_1 \wedge p_2\}$ in Example 1 above. Clearly, p_1 is a necessary condition of q on $\{p_1\}$. It's a strongest one because given any model of T_1, if it satisfies p_1, then we can modify it into another model of T_1 and q while preserving the truth value of p_1. However, although it is also a necessary condition of q on $\{p_1, p_2\}$, it is not the strongest one this time, because if $M \models p_1 \wedge \neg p_2 \wedge \neg q$, then $M \models T_1 \cup \{p_1\}$, but it cannot be modified into a model of q and T_1 without changing the truth value of p_2.

We can similarly define the notion of weakest sufficient conditions:

Definition 2 *Let T be a theory, P a set of propositions in T, and q a proposition in T but not in P. A formula ψ of P is said to be a* sufficient condition *of q on P under T if $T \models \psi \supset q$. It is said to be a* weakest sufficient condition *if it is a sufficient condition, and for any other sufficient condition ψ', we have that $T \models \psi' \supset \psi$.*

Strongest necessary and weakest sufficient conditions are in fact dual conditions. The easiest way to make this dual relationship precise is to extend these conditions from propositions to arbitrary formulas and to show that for any formula A, a formula φ is a strongest necessary (weakest sufficient) condition of A iff $\neg\varphi$ is a weakest sufficient (strongest necessary) condition of $\neg A$. This will be done in section 4.

We now relate these two conditions to the notion of definability. We say that a theory T *defines a proposition q* on a set P of propositions iff there is a formula φ of P such that $T \models q \equiv \varphi$. The following proposition is straightforward:

Proposition 2 *A theory T defines a proposition q on P iff $T \models \varphi \supset \phi$, where φ is any strongest necessary condition of q on P and ϕ any weakest sufficient*

condition of q on P, both under the theory T.

Notice that when the condition $T \models \varphi \supset \phi$ holds, we have that $T \models \varphi \equiv \phi$ and $T \models q \equiv \varphi$. So this proposition reduces the problem of computing definability to that of computing strongest necessary and weakest sufficient conditions. An advantage of working with the latter is that the two conditions always exist and are unique up to logical equivalence under any given theory. Furthermore, they are useful to have even when they do not yield a definition of a proposition. For instance, in logic of programs, Dijkstra's notion of weakest preconditions is a special case of our notion of weakest sufficient conditions, and as shown in [1, 4], these conditions provide a good way to reason about the indeterminate effects of actions for which successor state axioms (definitions) do not exist.

4 Strongest necessary and weakest sufficient conditions of a formula

Our notion of strongest necessary and weakest sufficient conditions can be extended from a proposition to an arbitrary formula.

Definition 3 *Let T be a propositional theory, α a formula, and P a set of propositions in $T \cup \{\alpha\}$. A formula φ of P is said to be a necessary condition of α on P iff $T \models \alpha \supset \varphi$. It is said to be a strongest necessary condition if for any other necessary condition φ', we have that $T \models \varphi \supset \varphi'$. Sufficient conditions and weakest sufficient conditions are defined similarly.*

Computing the strongest necessary and the weakest sufficient conditions of a formula can be reduced to that of a proposition, as the following proposition shows.

Proposition 3 *Let T, P, and α be as in Definition 3. A formula φ of P is the strongest necessary (weakest sufficient) condition of α on P under the theory T iff it is the strongest necessary (weakest sufficient) condition of q on P under the theory $T' = T \cup \{q \equiv \alpha\}$, where q is a new proposition not in T and α.*

Proof: We prove this for SNC. The case for WSC is similar. The "if" part: suppose φ is the SNC of α. We show that it is also an SNC of q under T'. It is easy to see that it is an NC of q: $T' \models q \supset \varphi$. Let φ' be any other necessary condition: $T' \models q \supset \varphi'$. Thus $T' \models \alpha \supset \varphi'$. Since q does not occur in α and φ', by Lemma 1, we have that $forget(T', q) \models \alpha \supset \varphi'$. But $forget(T', q)$ is equivalent to T since q does not occur in T and α. So φ' is also a necessary condition of α on P under T. Thus $T \models \varphi \supset \varphi'$. So $T' \models \varphi \supset \varphi'$.

The "only if" part: suppose φ is the SNC of q under T'. First of all, it is an NC: $T' \models q \supset \varphi$. Similar to the reasoning above, we have $T \models \alpha \supset \varphi$. So φ is also an NC of α on P under T. Now if φ' is any other NC of α under T: $T \models \alpha \supset \varphi'$. Then it is also an NC of q under T'. Thus by our assumption that φ is the SNC of q, we have that $T' \models \varphi \supset \varphi'$. Thus $T \models \varphi \supset \varphi'$ since q does not occur in T. ■

Although not necessary in principle, the notion of sufficient and necessary conditions of a formula is very useful in expressing many properties. The following are some interesting ones.

The first one says that, as expected, if two formulas are equivalent under T, then they have the same strongest necessary and weakest sufficient conditions:

Proposition 4 *If $T \models \alpha \equiv \beta$, then for any set of propositions P, a formula φ is a strongest necessary (weakest sufficient) condition of α on P iff it is a strongest necessary (weakest sufficient) condition of β on P.*

Proof: Suppose φ is the SNC of α on P under T. Then it is NC of β as well. Now let φ' be any SC of β: $T \models \beta \supset \varphi'$. Then we have $T \models \alpha \supset \varphi'$, so $T \models \varphi \supset \varphi'$. Therefore φ is also an SNC of β on P. The proof that if φ is an SNC of β, then it is also an SNC of α is symmetric.

The proof for WSC is similar. ■

The following proposition makes precise the dual relation between strongest necessary and weakest sufficient conditions:

Proposition 5 *A formula φ is the strongest necessary (weakest sufficient) condition of q iff $\neg\varphi$ is the weakest sufficient (strongest necessary) condition of $\neg q$, where all the conditions are on a common set of propositions and under a common theory.*

Proof: Suppose φ is the SNC of q. Then $T \models q \supset \varphi$. Thus $T \models \neg\varphi \supset \neg q$. So $\neg\varphi$ is a SC of $\neg q$. Suppose φ' is any other SC of $\neg q$: $T \models \varphi' \supset \neg q$. Then $T \models q \supset \neg\varphi'$. Thus $T \models \varphi \supset \neg\varphi'$. So $T \models \varphi' \supset \neg\varphi$. This proves that $\neg\varphi$ is the WSC of $\neg q$.

The proof of the other parts of the proposition is similar. ■

Proposition 6 *If φ_1, ..., φ_k are the strongest necessary conditions of $\alpha_1, ..., \alpha_k$, respectively, then $\varphi = \varphi_1 \vee \cdots \vee \varphi_k$ is the strongest necessary condition of*

$\alpha_1 \vee \cdots \vee \alpha_k$, *where all the conditions are on a common set of propositions and under a common theory.*

Proof: First of all, φ is an NC: $T \models \alpha \supset \varphi$. We show that it is the SNC using Theorem 1, generalized to arbitrary formula using Proposition 3: we show that if $M \models \varphi$ and $M \models T$, then there is a M' such that $M' \models T$, $M' \models \alpha$, and $M'(P) = M(P)$. Since $M \models \varphi$, there must be an $1 \leq i \leq k$ such that $M \models \varphi_i$. By Theorem 1 and Proposition 3, there is an M' such that $M'(P) = M(P)$, $M' \models T$, and $M' \models \alpha_i$, thus $M' \models \alpha$. ■

Symmetrically, for weakest sufficient conditions, we have:

Proposition 7 *If φ_1, ..., φ_k are the weakest sufficient conditions of $\alpha_1, ..., \alpha_k$, respectively, then $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_k$ is the weakest sufficient condition of $\alpha_1 \wedge \cdots \wedge \alpha_k$, where all the conditions are on a common set of propositions and under a common theory.*

This proposition is particularly useful in planning when we do not have a successor state axiom for every fluent: given a conjunctive goal $g_1 \wedge \cdots \wedge g_n$, to achieve it using action A, we need to compute the weakest sufficient condition of this conjunction, and reduce the conjunctive goal to it. By this proposition, instead of having to compute the weakest sufficient condition for every possible conjunction, it is enough to compute the condition for each conjunct ahead of time.

5 Computing strongest necessary and weakest sufficient conditions

There are several ways of computing these two conditions. We begin with prime implicates [15]. As with most reasoning tasks, strongest necessary and weakest sufficient conditions can be easily computed using prime implicates.

Proposition 8 *Let T be a theory, q a proposition, and P a set of propositions. Let Π be the set of prime implicates of T that mention only propositions in $P \cup \{q\}$.*

1. *The strongest necessary condition of q on P under T is equivalent to the following conjunction:*

$$\bigwedge\{C \mid \neg q \vee C \in \Pi\},$$

where if there is no such prime implicate in Π, then the conjunction is considered to be true, and if $\neg q \in \Pi$, then the conjunction is considered to be false.

2. *The weakest sufficient condition of q on P under T is equivalent to the following disjunction:*

$$\bigvee\{\neg C \mid q \vee C \in \Pi\},$$

where if there is no such prime implicate in Π, then the disjunction is considered to be false, and if $q \in \Pi$, then the conjunction is considered to be true.

Proof: We prove this proposition for SNC. Let W be the conjunction in the proposition. Clearly, $T \models q \supset W$, so W is an NC. Now let ψ be any other NC: $T \models q \supset \psi$, and Γ the set of prime implicates of ψ. Since ψ does not mention q, the following set

$$\Gamma' = \{\neg q \vee C \mid C \in \Gamma\}$$

is the set of prime implicates of $q \supset \psi$, where if $C = true$ (i.e., ψ is a tautology), then $\neg q \vee C$ is $true$, and if $C = false$ (i.e., ψ is not satisfiable), then $\neg q \vee C$ is $\neg q$. Therefore for each $\neg q \vee C$ in Γ', there is a prime implicate $\neg q \vee C'$ in Π such that C is subsumed by C'. Thus each prime implicate in Γ is subsumed by a conjunct in W. So $W \models \psi$. ■

However, computing the strongest necessary (similarly weakest sufficient) condition of a proposition using prime implicates is almost always a bad idea, unless one wants these conditions for all propositions on all possible sets of propositions and wants them in the form of prime implicates. This is because, in general, there is no viable way of computing the set Π in Proposition 8 short of computing the set of all prime implicates of a theory. But once having the set of prime implicates, one can just "read" out these two conditions for any proposition using Proposition 8.

A better way is in terms of the notion of forgetting that we defined in Section 2, by using the following theorem:[2]

Theorem 2 *Let T be a theory, P a set of propositions, and q a proposition in T but not in P. Let P' be the set of propositions that are in T but not in $P \cup \{q\}$. Then we have:*

1. *The strongest necessary condition of q on P is $forget(T(q/true); P')$.*
2. *The weakest sufficient condition of q on P is $\neg forget(T(q/false); P')$.*

[2] Lang and Marquis [5] showed that a proposition q is definable on P under T iff $T \models q \equiv forget(T(q/true); P')$. By Proposition 2, our theorem generalizes this result.

Proof: We prove the case for SNC. The case for WSC is similar. First of all, $forget(T(q/true); P')$ is an NC: $T \models forget(T; P')$, so $T \models q \supset forget(T; P')$, thus $T \models q \supset forget(T(q/true); P')$ because $q \notin P'$. It is the strongest: suppose α is an NC: α is a formula of P, and $T \models q \supset \alpha$. Thus, by Lemma 1, we have that $forget(T; P') \models q \supset \alpha$, Therefore $q \models forget(T; P') \supset \alpha$. So $q \models forget(T(q/true); P') \supset \alpha$. Thus $\models forget(T(q/true); P') \supset \alpha$ because q does not occur in $forget(T(q/true); P') \supset \alpha$. ■

We illustrate the use of this theorem using the following theory

$$T = \{q \supset p_1 \vee p_2\}.$$

According to the theorem, the strongest necessary condition of q on $\{p_1, p_2\}$ is $forget(T(q/true); \emptyset)$, which is $T(q/true)$ and equivalent to $p_1 \vee p_2$, the same as given in Example 1. The strongest necessary condition of q on $\{p_1\}$ is $forget(T(q/true); \{p_2\})$, which is equivalent to $forget(p_1 \vee p_2; \{p_2\})$, which by definition is $(p_1 \vee true) \vee (p_1 \vee false)$, thus equivalent to $true$.

By its definition, for any formula φ and proposition p, $forget(\varphi; p)$ returns a formula that is twice the size as that of φ; and for a set P of propositions, $forget(\varphi; P)$ returns a formula whose size is 2^n times the size of φ, where n is the size of P. Indeed, computing forgetting in the worst case is expensive. For instance, it is easy to see that a formula is satisfiable iff forgetting all propositions in it returns $true$, and a formula is not satisfiable iff forgetting all propositions in it returns $false$. However, $forget(\varphi; p)$ can often be simplified. For instance, if φ is in disjunctive normal form, then $forget(\varphi; p)$ can be computed efficiently to yield a formula that is shorter than φ: it is the result of replacing both p and $\neg p$ in φ by $true$. More generally, as Darwiche [3] showed, if φ is what he called decomposable negation normal form, then $forget(\varphi; p)$ can be computed efficiently.

In this paper, we consider two ways of computing forgetting: directly using its definition or through (again) prime implicates. They are embodied below as Algorithms 1 and 2 for computing strongest necessary conditions. The ones for weakest sufficient conditions are similar.

Input A set T of axioms, a set P of propositions, and a proposition q not in P.

Output A formula of P that is the strongest necessary condition of q on P under T.

Algorithm 1

1. Let $T_1 = \{\varphi \mid \varphi \in T$ is a sentence of $P\}$, and $T_2 = T - T1$. Replace q in T_2 by $true$, and transform the resulting theory into a *minimal set of clauses* C (see below for a definition). Similarly, transform T_1 into a minimal set of clauses C_0.
2. Let P' be the set of propositions in C but not in P.
3. If P' is empty, then go to post-processing, otherwise select a proposition p in P', and delete it from P'.
4. Transform $C(p/true) \vee C(p/false)$ into a minimal set of clauses, assign it to C, and go back to step 3.
5. *Post-processing:* although the resulting C is a strongest necessary condition of q, it is often an unwieldy set of clauses; it can be simplified as follows:
   - eliminate those clauses in C that are subsumed by one of the clauses in C_0;
   - for each clause α in C, transform $(C - \{\alpha\}) \cup C_0$ into a minimal set of clauses C_α, and eliminate α from C if it is subsumed by one of the clauses in C_α.

Where a minimal set of clauses is one such that:

- all unit clauses are resolved;
- none of the clauses is subsumed by any other clause in the set.

Our experience with this algorithm has been that it spends the bulk of the time on computing minimal sets of clauses in step 4. But if we do not require a set of clauses to be minimal, then the program quickly runs out of space.

Notice that according to our definition, for a set of clauses C to be a minimal one, only unit clauses in it need to be resolved. So our notion of minimality is much weaker than that of prime implicates. For comparison, let us consider computing forgetting using prime implicates.

The following algorithm makes use of the fact that for any φ and P, $forget(\varphi; P)$ is equivalent to the conjunction of prime implicates of φ that do not mention any propositions in P.

Algorithm 2

1. Let $T_1 = \{\varphi \mid \varphi \in T$ is a sentence of $P\}$, and $T_2 = T - T1$. Transform T_1 into a minimal set of clauses C_0.
2. Generate the prime implicates of $T_2(q/true)$ using Tison's procedure [15], and let C be the set of those prime implicates that mention only propositions in P.
3. Go to post-processing as in Algorithm 1.

We have implemented the above two algorithms in SWI-Prolog[3], and run them on both random 3CNFs[4] and action theories used for computing successor state axioms in various action domains [8]. It turned out that Algorithm 2 is always slower than Algorithm 1, with on average a slow down of between 10 to 15 times. The reason is that computing prime implicates is expensive. Our notion of minimal sets of clauses seems to serve our purpose well here in cutting down the number and sizes of clauses without incurring too much overhead in computing them.

Notice that in step 4 of Algorithm 1, after we have chosen a proposition to forget, we compute immediately the result of forgetting this proposition, $C(p/true) \vee C(p/false)$, by a set of minimal clauses. This is like breadth-first search, and as we have mentioned, is the bottleneck of this algorithm. Alternatively, we could work on each disjunct separately, and compute disjunctions only after all those propositions that need to be forgotten have been eliminated. As it turned out, this variant of Algorithm 1 performed far better than Algorithm 1 on random 3CNFs. For random 3CNFs with 50 variables and 215 clauses, this variant of Algorithm 1 spent on average 15 minutes[5] to return a strongest necessary condition of a proposition on a set of 10 other propositions, while Algorithm 1 did not return a solution after running overnight. However, this variant of Algorithm 1 is about 20% slower on action theories. The key is with the disjunction $C(p/true) \vee C(p/false)$: it can be simplified a lot for many benchmark action theories, but not much for random theories.

Regardless of which approach one uses, we have found that sometimes, it is a lot easier to compute the strongest necessary condition than the weakest sufficient condition, and sometimes, it is the other way around. For instance, we have found that for many action theories, strongest necessary conditions are a lot easier to compute than weakest sufficient conditions. When one condition is much easier to compute, the following proposition will be very useful in computing the other one.

Proposition 9 *Let T be a theory, q a proposition, and P a set of propositions.*

1. *If φ is a necessary condition of q on P under T, and ψ a weakest sufficient condition of q on P under $T \cup \{\varphi\}$, then $\varphi \wedge \psi$ is a weakest sufficient condition of q on P under T.*
2. *If ψ is a sufficient condition of q on P under T, and φ a strongest necessary condition of q on P under $T \cup \{\neg\psi\}$, then $\varphi \vee \psi$ is a strongest necessary condition of q on P under T.*

Proof:

1. First of all, $\varphi \wedge \psi$ is an SC: $T \cup \{\varphi\} \models \psi \supset q$, so $T \models \varphi \wedge \psi \supset q$. It is the weakest: suppose α is an SC: $T \models \alpha \supset q$. We need to show that $T \models \alpha \supset \varphi \wedge \psi$. Since $T \models q \supset \varphi$. We have $T \models \alpha \supset \varphi$. But α is also an SC of q under $T \cup \{\varphi\}$, so $T \cup \{\varphi\} \models \alpha \supset \psi$ because ψ is the WSC of q under $T \cup \{\varphi\}$. Therefore $T \models \alpha \supset (\varphi \wedge \psi)$.
2. Suppose φ' is an NC of q under T: $T \models q \supset \varphi'$. From $T \models \psi \supset q$, we have $T \models \psi \supset \varphi'$. From $T \cup \{\neg\psi\} \models q \supset \varphi$ and $T \cup \{\neg\psi\} \models q \supset \varphi'$, we have $T \cup \{\neg\psi\} \models \varphi \supset \varphi'$. Thus $T \models \neg\psi \wedge \varphi \supset \varphi'$, this together with $T \models \psi \supset \varphi'$, we have $T \models \varphi \supset \varphi'$.

■

We illustrate this phenomenon for the problem of generating successor state axioms in a robot domain. To simplify our presentation, we assume that there is just one object in this domain, and the robot can carry the object around by moving from one location to another. Now suppose that $move(1,2)$ stands for the action that the robot moves from location 1 to location 2, and that we have the following propositions:

- at_i - initially, the object is at location i, for each location i.
- $at1_i$ - after the action $move(1,2)$, the object is at location i.
- atR_i - initially, the robot is at location i.
- $atR1_i$ - after the action, the robot is at location i.

[3]SWI-Prolog is developed by Jan Wielemaker at the University of Amsterdam

[4]Generated using a program by Kautz and Selman.

[5]On a SPARC Ultra2 machine running SWI-Prolog.

- h - initially, the robot is holding the object.
- $h1$ - after the action, the robot is holding the object.

The background theory is then the following set of axioms:

$$atR_1 \wedge atR1_2,$$
$$atR1_i \equiv atR_i \wedge \neg \bigvee_{j \neq i} atR1_j,$$
$$at1_i \equiv [atR1_i \wedge h1] \vee [at_i \wedge \neg \bigvee_{j \neq i} at1_j],$$
$$h1 \equiv h,$$
$$atR_i \supset \neg atR_j, \text{ for any } j \neq i$$
$$at_i \supset \neg at_j, \text{ for any } j \neq i$$

where the disjunctions are ranged over all locations. These axioms are generated from the direct effect action axiom: after $move(1,2)$, the robot will be at location 2, and domain constraints such as "the object and the robot can be at only one location at any given time" and "if the robot is holding the object, then the robot and the object are always at the same location" using a predicate completion procedure in [7]. More details about this procedure, and the way the above axioms are generated from a causal theory can be found in (Lin [8]). Now let q be $at1_2$, and P the set of propositions about the initial situation:

$$P = \{h\} \cup \{at_i |\ i \text{ is a location}\} \cup \{atR_i |\ i \text{ is a location}\}.$$

Then the weakest sufficient condition of q on P is the weakest condition about the initial situation that would ensure that the package is at location 2 after the action is performed, and the strongest necessary condition is the strongest conclusion that one can infer about the initial situation once one knows that the package is at location 2 after the action is performed.

Figure 1 compares the performance of computing the weakest sufficient condition of q on P using Algorithm 1 with or without the aid of Proposition 9. In the figure, the x-axis is the number of locations, and y-axis the CPU run time in seconds on a SPARC Ultra 2. The line labeled by *snc* corresponds to the strongest necessary condition of q on P,[6] the one labeled by *wsc* the weakest sufficient condition of q on P,[7] and *wsc*1 the weakest sufficient condition of q on P once the strongest sufficient condition has been added to the theory[8] according to Proposition 9. The speedup of

[6]For which the program outputs $h \vee at_2$ no matter how many locations there are in the domain.

[7]For which the program outputs $h \vee at_2$.

[8]For which the program outputs *true*.

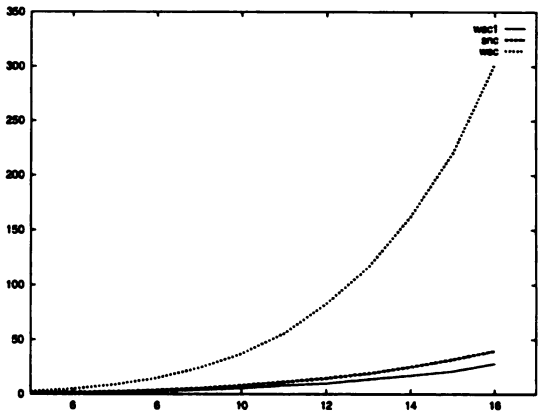


Figure 1: A robot domain

*wsc*1 over *wsc* is quite obvious, and similar speedup is also achieved for other propositions and other action domains that we have experimented with, which include most of the benchmark planning domain in McDermott's PDDL library [11].

6 Concluding remarks and future work

We have proposed a notion of strongest necessary and weakest sufficient conditions of a proposition, and considered ways of computing them. We believe these conditions have many potential applications in various areas including abduction and reasoning about actions.

There are several directions for future work. One of them is to extend the results here to the first-order case. This can be a difficult task. For instance, a result in [9] shows that forgetting in the first-order case cannot in general be expressible in first-order logic. As a consequence, we expect that strongest necessary conditions of a proposition under a first-order theory cannot in general be expressible in first-order logic either. It seems that the best hope for dealing with the first-order case is to first reduce it to the propositional case, and then try to learn a first-order description from a set of propositional ones.

Acknowledgments

I would like to thank Maurice Pagnucco for his comments on an earlier version of this paper. This work was supported in part by the Research Grants Council of Hong Kong under Competitive Earmarked Research Grants HKUST6091/97E and HKUST6145/98E.

References

[1] M. Bjäreland and L. Karlsson. Reasoning by regression: pre- and postdiction procedures for logics of action and change with nondeterminism. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97), IJCAI Inc. Distributed by Morgan Kaufmann, San Mateo, CA.*, pages 1420–1425, 1997.

[2] G. Boole. *An Investigation of the Laws of Thought.* Walton, London, 1854. (Reprinted by Dover Books, New York, 1954).

[3] A. Darwiche. Compiling knowledge into decomposable negation normal form. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99), IJCAI Inc. Distributed by Morgan Kaufmann, San Mateo, CA.*, 1999.

[4] E. W. Dijkstra and C. S. Scholten. *Predicate Calculus and Program Semantics.* Springer-Verlag, New York, 1990.

[5] J. Lang and P. Marquis. Two forms of dependence in propositional logic: controllability and definability. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98), AAAI Press, Menlo Park, CA.*, 1998.

[6] C. I. Lewis. *A Survey of Symbolic Logic.* University of California Press, Berkeley, 1918. (Reprinted by Dover Pub's., Inc., New York, 1960. Chapter II in The Classic or Boole-Schroder Algebra of Logic).

[7] F. Lin. Embracing causality in specifying the indirect effects of actions. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), IJCAI Inc. Distributed by Morgan Kaufmann, San Mateo, CA.*, pages 1985–1993, 1995.

[8] F. Lin. From causal theories to successor state axioms: bridging the gap between nonmonotonic action theories and STRIPS-like formalisms. In *http://www.cs.ust.hk/faculty/flin*, 1999. Submitted.

[9] F. Lin and R. Reiter. How to progress a database. *Artificial Intelligence*, (92)1-2:131–167, 1997.

[10] F. Lin and R. Reiter. Forget it! In R. Greiner and D. Subramanian, editors, *Working Notes of AAAI Fall Symposium on Relevance*, pages 154–159. The American Association for Artificial Intelligence, Menlo Park, CA, November 1994. Also available at `http://www.cs.toronto.edu/cogrobo/forgetting.ps.Z`.

[11] D. McDermott et al. (AIPS-98 Planning Competition Committee). PDDL – the planning domain definition language. Technical Report TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

[12] J. Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–709, 1995.

[13] R. Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 418–420. Academic Press, San Diego, CA, 1991.

[14] H. Simon. Causal ordering and identifiability. In W. Hood and T. Koopmans, editors, *Studies in Econometric Method.* John Wiley and Sons, New York, 1953.

[15] P. Tison. Generalized consensus theory and application to the minimization of boolean functions. *IEEE Trans. on Electronic Computers*, EC-16/4:446–456, 1967.

[16] A. Weber. Updating propositional formulas. In *Proceedings First Conference on Expert Database Systems*, pages 487–500, 1986.

Containment of Conjunctive Regular Path Queries with Inverse

Diego Calvanese[1], Giuseppe De Giacomo[1], Maurizio Lenzerini[1], Moshe Y. Vardi[2]

[1] Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
lastname@dis.uniroma1.it
http://www.dis.uniroma1.it/~*lastname*

[2] Department of Computer Science
Rice University, P.O. Box 1892
Houston, TX 77251-1892, U.S.A.
vardi@cs.rice.edu
http://www.cs.rice.edu/~vardi

Abstract

Reasoning on queries is a basic problem both in knowledge representation and databases. A fundamental form of reasoning on queries is checking containment, i.e., verifying whether one query yields necessarily a subset of the result of another query. Query containment is crucial in several contexts, such as query optimization, knowledge base verification, information integration, database integrity checking, and cooperative answering.

In this paper we address the problem of query containment in the context of semistructured knowledge bases, where the basic querying mechanism, namely regular path queries, asks for all pairs of objects that are connected by a path conforming to a regular expression. We consider conjunctive regular path queries with inverse, which extend regular path queries with the possibility of using both the inverse of binary relations, and conjunctions of atoms, where each atom specifies that one regular path query with inverse holds between two variables. We present a novel technique to check containment of queries in this class, based on the use of two-way finite automata. The technique shows the power of two-way automata in dealing with the inverse operator and with the variables in the queries. We also characterize the computational complexity of both the proposed algorithm and the problem.

1 INTRODUCTION

Querying is the fundamental mechanism for extracting information from a knowledge base. The basic reasoning task associated to querying is query answering, which amounts to compute the information to be returned as result of a query. However, there are other reasoning services involving queries that knowledge representation systems should support. One of the most important is checking containment, i.e., verifying whether one query yields necessarily a subset of the result of another one. Query containment is crucial in several contexts, such as query optimization, knowledge base verification, information integration, integrity checking, and cooperative answering.

Query optimization aims at improving the efficiency of query answering, and largely benefits from the possibility of performing various kinds of comparisons between query expressions. In particular, query containment checks are useful to recognize equivalent subexpressions, to avoid computing results already available, to recognize the possibility of using materialized views, and to use integrity constraints to speed-up query processing (Levy & Sagiv, 1995; Chaudhuri, Krishnamurthy, Potarnianos, & Shim, 1995; Widom, 1995; Adali, Candan, Papakonstantinou, & Subrahmanian, 1996; Buneman, Davidson, Hillebrand, & Suciu, 1996; Fernandez & Suciu, 1998; Milo & Suciu, 1999).

Recently, it has been shown that query containment is relevant for the task of knowledge base verification (Levy & Rousset, 1997). In (Levy & Rousset, 1998b), the problem of verifying whether a knowledge base produces the correct set of output for any set of input is solved by means of a method that exploits the ability of checking query containment.

One of the major issues in information integration (Fensel, Knoblock, Kushmerick, & Rousset, 1999) is to reformulate a query expressed over a unified domain representation in terms of the local sources. Several recent papers point out that query containment is essential for this purpose (Calvanese, De Giacomo, Lenzerini, Nardi, & Rosati, 1998; Levy, Rajaraman,

& Ordille, 1996; Knoblock & Levy, 1995; Friedman, Levy, & Millstein, 1999). Also, many information integration applications are developed on the web, where data are expressed using XML-like languages (Bray, Paoli, & Sperberg-McQueen, 1998; Calvanese, De Giacomo, & Lenzerini, 1999) and semistructured models (Buneman, 1997; Florescu, Levy, & Mendelzon, 1998a). This new scenario poses interesting challenges to both database and knowledge representation technologies, and query containment is one notable example of such challenges (Florescu, Levy, & Suciu, 1998b; Calvanese, De Giacomo, Lenzerini, & Vardi, 2000).

Besides the above described applications, query containment is crucial in integrity constraint checking (Gupta & Ullman, 1992; Fernandez, Florescu, Levy, & Suciu, 1999), cooperative answering (Motro, 1996), and in general in knowledge representation systems based on description logics and conceptual graphs, where it comes in the form of subsumption check, and is at the heart of all relevant reasoning tasks (Donini, Lenzerini, Nardi, & Schaerf, 1996; Eklund, Nagle, Nagle, & Gerholz, 1992; Donini, Lenzerini, Nardi, & Schaerf, 1998; Levy & Rousset, 1998a).

Needless to say, query containment is undecidable if we do not limit the expressive power of the query language. In fact, in knowledge representation suitable query languages have been designed for retaining decidability. The same is true in databases, where the notion of conjunctive query is the basic one in the investigation on reasoning on queries (Chandra & Merlin, 1977). A conjunctive query is simply a conjunction of atoms, where each atom is built out from relation symbols and (existentially quantified) variables, and correspond to a single rule in non-recursive datalog.

Most of the results on query containment concern conjunctive queries and their extensions. In (Chandra & Merlin, 1977), NP-completeness has been established for conjunctive queries, in (Klug, 1988; van der Meyden, 1992), Π_2^p-completeness of containment of conjunctive queries with inequalities was proved, and in (Sagiv & Yannakakis, 1980) the case of queries with the union and difference operators was studied. For various classes of datalog queries with inequalities, decidability and undecidability results were presented in (Chaudhuri & Vardi, 1992) and (van der Meyden, 1992), respectively. Other papers consider the case of conjunctive query containment in the presence of various types of constraints (Aho, Sagiv, & Ullman, 1979; David S. Johnson, 1984; Chan, 1992; Levy & Rousset, 1996; Levy & Suciu, 1997), or in knowledge representation systems integrating datalog and description logics (Levy & Rousset, 1998a).

In this paper we address the problem of query containment in the context of a general form of knowledge bases, called semistructured knowledge bases. Our goal is to capture the essential features of knowledge bases as found in semantic networks, description logics, conceptual graphs, and in databases, both traditional and semistructured. For this purpose, we conceive a knowledge base as a labeled graph, where nodes represent objects, and a labeled edge between two nodes represents the fact that the binary relation denoted by the label holds for the objects.

In our framework, the basic querying mechanism is the one of regular path queries (Buneman, 1997; Calvanese, De Giacomo, Lenzerini, & Vardi, 1999; Abiteboul, Buneman, & Suciu, 1999), which ask for all pairs of objects that are connected by a path conforming to a regular expression. Regular path queries are extremely useful for expressing complex navigations in a graph. In particular, union and transitive closure are crucial when we do not have a complete knowledge of the structure of the knowledge base.

In this work, we consider *conjunctive regular path queries with inverse*, which extend regular path queries with the possibility of using both the inverse of binary relations, and conjunctions of atoms, where each atom specifies that one regular path query with inverse holds between two variables. Notably, several authors argue that these kinds of extensions are essential for making regular path queries useful in real settings (see for example (Buneman, 1997; Buneman et al., 1996; Milo & Suciu, 1999)). Conjunctive regular path queries have been studied in (Florescu et al., 1998b), where an EXPSPACE algorithm for query containment in this class is presented. However, no lower bound is provided for the problem, and, moreover, the method does not seem easily generalizable to take into account the inverse operator. The case with the inverse operator is implicitly addressed in (Calvanese, De Giacomo, & Lenzerini, 1998), where an 2EXPTIME algorithm is proposed for query containment. However, the framework investigated in (Calvanese et al., 1998) includes a rich set of constraints, and is not suited for a precise characterization of containment of conjunctive regular path queries with inverse.

We present a novel technique to check containment of queries in this class, based on the use of two-way finite automata. Differently from standard finite state automata, two-way automata are equipped with a head that can move back and forth on the input string. A transition of these kinds of automata indicates not only the new state, but also whether the head should move left, right, or stay in place. Our technique shows the

power of two-way automata in dealing with the inverse operator and with the variables in conjunctive queries. In particular, we describe an algorithm that checks containment of two queries by checking nonemptiness of a two-way automaton constructed from the two queries. The algorithm works in exponential space, and therefore has the same worst-case complexity as the best algorithm known for the case of conjunctive regular path queries without inverse (Florescu et al., 1998b).

We also prove an EXPSPACE lower bound for the computational complexity of the problem, thus demonstrating that our method is essentially optimal. Interestingly, the lower bound holds even if we disregard the inverse operator, and therefore provides the solution to the open problem of whether containment of conjunctive regular path queries could be done in PSPACE (Florescu et al., 1998b). Besides the specific result, our method provides the basis for using two-way automata in reasoning on complex queries, and can be adapted to more general forms of queries (e.g., unions of conjunctive queries) and reasoning tasks (e.g., query rewriting), as well as to other formalisms in Artificial Intelligence (e.g., temporal and dynamic logics with a backward modality, and action theories with converse).

2 KNOWLEDGE BASES AND QUERIES

We consider a *semistructured knowledge base* (KB) $\mathcal{K}$ as an edge-labeled graph $(\mathcal{D}, \mathcal{E})$, where $\mathcal{D}$ is the set of nodes, and $\mathcal{E}$ is the set of edges labeled with elements of an alphabet Σ'. A node represents an object, and an edge between nodes x and y labeled p represents the fact that the binary relation p holds for the pair (x, y). We denote an edge from x to y labeled by p with $x \xrightarrow{p} y$.

The basic querying mechanism on a KB is that of *regular path queries* (RPQs). An RPQ R is expressed as a regular expression or a finite automaton, and computes the set of pairs of nodes of the KB connected by a path that conforms to the regular language $L(R)$ defined by R. As we said in the introduction, we consider queries that extend regular path queries with both the inverse operator, and the possibility of using conjunctions and variables.

Formally, let $\Sigma = \Sigma' \cup \{p^- \mid p \in \Sigma'\}$ be the alphabet including a new symbol p^- for each p in Σ'. Intuitively, p^- denotes the inverse of the binary relation p. If $r \in \Sigma$, then we use r^- to mean the *inverse* of r, i.e., if r is p, then r^- is p^-, and if r is p^-, then r^- is p.

Regular path queries with inverse (RPQIs) are expressed by means of regular expressions or finite automata over Σ. Thus, in contrast with RPQs, RPQIs may use also the inverse p^- of p, for each $p \in \Sigma'$. When evaluated over a KB $\mathcal{K}$, an RPQI E computes the set $ans(E, \mathcal{K})$ of pairs of nodes connected by a semipath that conforms to the regular language $L(E)$ defined by E. A *semipath* in $\mathcal{K}$ from x to y is a sequence of the form $(y_1, r_1, y_2, r_2, y_3, \ldots, y_q, r_q, y_{q+1})$, where $q \geq 0$, $y_1 = x$, $y_{q+1} = y$, and for each y_i, r_i, y_{i+1}, either $y_i \xrightarrow{r_i} y_{i+1}$ or $y_{i+1} \xrightarrow{r_i^-} y_i$ is in $\mathcal{K}$. The semipath *conforms to* E if $r_1 \cdots r_q \in L(E)$. The semipath is *simple* if each y_i, for $i \in \{2, \ldots, q\}$, is a node that does not occur elsewhere in the semipath.

Finally, we add to RPQIs the possibility of using conjunctions of atoms, where each atom specifies that one regular path query with inverse holds between two variables. More precisely, if Φ is an alphabet of variables, then a *conjunctive regular path query with inverse* (CRPQI) Q is a formula of the form

$$Q(x_1, \ldots, x_n) \leftarrow y_1\, E_1\, y_2 \wedge \cdots \wedge y_{2m-1}\, E_m\, y_{2m}$$

where $x_1, \ldots, x_n, y_1, \ldots, y_{2m}$ range over a set $\{u_1, \ldots, u_k\}$ of variables in Φ, each x_i, called a *distinguished variable*, is one of $y_1, \ldots, y_{2m}$, and $E_1, \ldots, E_m$ are RPQIs.

The *answer set* $ans(Q, \mathcal{K})$ to a CRPQI Q over a KB $\mathcal{K} = (\mathcal{D}, \mathcal{E})$ is the set of tuples $(d_1, \ldots, d_n)$ of nodes of $\mathcal{K}$ such that there is a total mapping σ from $\{u_1, \ldots, u_k\}$ to $\mathcal{D}$ with $\sigma(x_i) = d_i$ for every distinguished variable x_i of Q, and $(\sigma(y), \sigma(z)) \in ans(E, \mathcal{K})$ for every conjunct yEz in Q.

Example 1 Let us consider a KB of parental relationships. The CRPQI

$$\begin{aligned} Q(x_1, x_2) \leftarrow\ & \\ & x_1\ (\mathsf{father}^- \cdot \mathsf{father} \cup \mathsf{mother}^- \cdot \mathsf{mother})^+\ x_2 \wedge \\ & x_1\ (\mathsf{father} \cup \mathsf{mother})^* \cdot \mathsf{father}\ y \wedge \\ & x_2\ (\mathsf{father} \cup \mathsf{mother})^* \cdot \mathsf{mother}\ y \end{aligned}$$

returns all pairs of individuals (x_1, x_2) that are in the transitive closure of the sibling (including stepsibling) relation, and such that there is some individual y such that x_1 and x_2 have two descendants who are respectively the father and the mother of y. ∎

Given two CRPQIs Q_1 and Q_2, we say that Q_1 is *contained* in Q_2, written $Q_1 \subseteq Q_2$, if for every KB $\mathcal{K}$, $ans(Q_1, \mathcal{K}) \subseteq ans(Q_2, \mathcal{K})$. Obviously, $Q_1 \not\subseteq Q_2$ iff there is a *counterexample* KB to $Q_1 \subseteq Q_2$, i.e., a KB $\mathcal{K}$ with a tuple in $ans(Q_1, \mathcal{K})$ and not in $ans(Q_2, \mathcal{K})$.

Example 1 (cont.) It is possible to verify that the CRPQI

$$Q'(x_1, x_2) \leftarrow x_1\ \mathsf{father}^-\cdot\mathsf{father}\cdot\mathsf{mother}^-\cdot\mathsf{mother}\ x_2 \wedge x_1\ \mathsf{father}\cdot\mathsf{mother}^-\ x_2$$

is contained in Q. ∎

In order to characterize containment between CRPQIs, we introduce the notion of canonical KB. Let Q be the CRPQI

$$Q(x_1, \ldots, x_n) \leftarrow y_1\ E_1\ y_2 \wedge \cdots \wedge y_{2m-1}\ E_m\ y_{2m}$$

$\mathcal{K}$ be a KB, and ν be a total mapping from the variables $\{u_1, \ldots, u_k\}$ of Q to the nodes of $\mathcal{K}$. Then $\mathcal{K}$ is said to be *ν-canonical* for Q if:

- $\mathcal{K}$ constitutes of m simple semipaths, one for each conjunct of Q, which are node and edge disjoint, i.e., only start and end nodes can be shared between different semipaths.
- for $i \in \{1, \ldots, m\}$, the simple semipath associated to the conjunct $y_{2i-1}\ E_i\ y_{2i}$ connects the node $\nu(y_{2i-1})$ to the node $\nu(y_{2i})$, and conforms to E_i.

It is easy to see that, if $\mathcal{K}$ is ν-canonical for Q, then the tuple $(\nu(x_1), \ldots, \nu(x_n))$ belongs to $ans(Q, \mathcal{K})$, and therefore $ans(Q, \mathcal{K})$ is nonempty.

In the following, we assume that Q_h, for $h = \{1, 2\}$, are of the form

$$Q_h(x_1, \ldots, x_n) \leftarrow y_{h,1}\ E_{h,1}\ y_{h,2} \wedge \cdots \wedge y_{h,2m_h-1}\ E_{h,m_h}\ y_{h,2m_h}$$

i.e., Q_1 and Q_2 have the same distinguished variables $x_1, \ldots, x_n$, and the sets of non-distinguished variables of respectively Q_1 and Q_2 are disjoint. This assumption can be made without loss of generality, since we can rename variables and simulate equality between x and y by introducing $x\varepsilon y$ in the right hand side.

Let $\mathcal{K} = (\mathcal{D}, \mathcal{E})$ be a ν-canonical KB for Q_1. A total mapping μ from the variables of Q_2 to $\mathcal{D}$, is called a $(Q_2, \mathcal{K}, \nu)$-mapping if

- it maps distinguished variables of Q_2 into nodes of $\mathcal{K}$ corresponding to distinguished variables of Q_1, i.e., for each x_i, we have that $\mu(x_i) = \nu(x_i)$, and
- for all $j \in \{1, \ldots, m_2\}$, we have that $(\mu(y_{2,2j-1}), \mu(y_{2,2j})) \in ans(E_{2,j}, \mathcal{K})$.

Note that the existence of a $(Q_2, \mathcal{K}, \nu)$-mapping implies that $(\mu(x_1), \ldots, \mu(x_n)) \in ans(Q_2, \mathcal{K})$.

The following theorem provides an important characterization of containment between CRPQIs.

Theorem 2 *Let Q_1 and Q_2 be two CRPQIs. Then $Q_1 \not\subseteq Q_2$ iff there exists a KB $\mathcal{K}$ and a mapping ν from the variables of Q_1 to the nodes of $\mathcal{K}$ such that (i) $\mathcal{K}$ is ν-canonical for Q_1 and (ii) no $(Q_2, \mathcal{K}, \nu)$-mapping exists.*

Proof (sketch). For the if-part, it is easy to see that $\mathcal{K}$ is a counterexample to $Q_1 \subseteq Q_2$. For the only-if-part, it is possible to show that any counterexample can be transformed into a KB of the form stated in the theorem and that such KB is still a counterexample to $Q_1 \subseteq Q_2$. □

3 TWO-WAY AUTOMATA

A *two-way automaton* (Hopcroft & Ullman, 1979; Vardi, 1989) $A = (\Gamma, S, I, \delta, F)$ consists of an alphabet Γ, a finite set of states S, a set $I \subseteq S$ of initial states, a transition function $\delta : S \times \Sigma \to 2^{S \times \{-1,0,1\}}$, and a set $F \subseteq S$ of accepting states. Intuitively, a transition indicates both the new state of the automaton, and whether the head reading the input should move left (-1), right (1), or stay in place (0). If for all $s \in S$ and $a \in \Gamma$ we have that $\delta(s, a) \subseteq S \times \{1\}$, then the automaton is a traditional nondeterministic finite state automaton (also called *one-way automaton*).

A *configuration* of A is a pair consisting of a state and a position represented as a natural number. A *run* is a sequence of configurations. The sequence $((s_0, j_0), \ldots, (s_m, j_m))$ is a run of A on a word $w = a_0, \ldots, a_{n-1}$ in Γ^* if $s_0 \in I$, $j_0 = 0$, $j_m \leq n$, and for all $i \in \{0, \ldots, m-1\}$, we have that $0 \leq j_i < n$, and there is some $(t, k) \in \delta(s_i, a_{j_i})$ such that $s_{i+1} = t$ and $j_{i+1} = j_i + k$. This run is *accepting* if $j_m = n$ and $s_m \in F$. *A accepts w* if it has an accepting run on w. The set of words accepted by A is denoted $L(A)$.

It is well known that two-way automata define regular languages (Hopcroft & Ullman, 1979), and that, given a two-way automaton with n states, we can construct a one-way automaton with $O(2^{n \log n})$ states accepting the same language (Vardi, 1989).

We want to gain some intuition on how two-way automata capture computations of CRPQIs over KBs. To this end we show how a two-way automaton can be used for the fundamental task of verifying the nonemptiness of an RPQI over a given KB.

The basic idea that allows us to exploit automata is that we can represent special KBs by means of words. In particular, we consider KBs in which the domain $\mathcal{D}$ contains a fixed set $\mathcal{D}_0$ of nodes, and that are constituted by simple semipaths which are node and edge disjoint and such that the start and end nodes of each semipath are in $\mathcal{D}_0$. Each such KB $\mathcal{K} = (\mathcal{D}, \mathcal{E})$ is represented by a word $W_\mathcal{K}$ over the alphabet $\Sigma \cup \mathcal{D}_0 \cup \{\$\}$, which has the form

$$\$d_1 w_1 d_2 \$ d_3 w_2 d_4 \$ \cdots \$ d_{2m-1} w_m d_{2m} \$$$

where $d_1, \ldots, d_{2m}$ range over $\mathcal{D}_0$, $w_i \in \Sigma^+$, and the \$ acts as a separator. Specifically, $W_\mathcal{K}$ consists of one subword $d_{2i-1} w_i d_{2i}$, for each simple semipath in $\mathcal{K}$ from d_{2i-1} to d_{2i} conforming to w_i. Observe that we can represent the same KB by several words that differ in the direction considered for the semipaths and in the order in which the subwords corresponding to the semipaths appear.

Now, given an RPQI E, we build a two-way automaton A_E over the alphabet $\Sigma \cup \mathcal{D}_0 \cup \{\$\}$ such that A_E accepts a word $W_\mathcal{K}$ iff E has a nonempty answer on the KB $\mathcal{K}$ represented by $W_\mathcal{K}$. To do so we exploit the ability of two-way automata to:

- move on the word both forward and backward, which corresponds to traversing edges of the KB in both directions;
- "jump" from one position in the word representing a node to any other position (either preceding or succeeding) representing the same node.

These two capabilities ensure that the automaton evaluating the RPQI on the word simulates exactly the evaluation of the query on the KB.

To construct A_E, we assume that E is represented as a finite (one-way) automaton $E = (\Sigma, S, I, \delta, F)$ over the alphabet Σ. Then $A_E = (\Sigma_A, S_A, \{s_0\}, \delta_A, \{s_f\})$, where $\Sigma_A = \Sigma \cup \mathcal{D}_0 \cup \{\$\}$, $S_A = S \cup \{s_0\} \cup \{s^b \mid s \in S\} \cup S \times \mathcal{D}_0$, and δ_A is defined as follows:

1. $(s_0, 1) \in \delta_A(s_0, \ell)$, for each $\ell \in \Sigma_A$, and also $(s, 1) \in \delta_A(s_0, \ell)$ for each $s \in I$. These transitions place the head of the automaton in some randomly chosen position of the input string and set the state of the automaton to some randomly chosen initial state of E.
2. $(s^b, -1) \in \delta_A(s, \ell)$, for each $s \in S$ and $\ell \in \Sigma \cup \mathcal{D}_0$. At any point such transition makes the automaton ready to scan one step backward by placing it in "backward mode".
3. $(s_2, 1) \in \delta_A(s_1, r)$ and $(s_2, 0) \in \delta_A(s_1^b, r^-)$, for each transition $s_2 \in \delta(s_1, r)$ of E. These transitions correspond to the transitions of E which are performed forward or backward according to the current "scanning mode".
4. for each $s \in S$ and each $d \in \mathcal{D}_0$

$$\begin{array}{rcll} ((s,d),0) & \in & \delta_A(s,d) & \\ ((s,d),0) & \in & \delta_A(s^b,d) & \\ ((s,d),1) & \in & \delta_A((s,d),\ell), & \text{for each } \ell \in \Sigma_A \\ ((s,d),-1) & \in & \delta_A((s,d),\ell), & \text{for each } \ell \in \Sigma_A \\ (s,0) & \in & \delta_A((s,d),d) & \\ (s,1) & \in & \delta_A(s,d) & \end{array}$$

   Whenever the automaton reaches a symbol representing a node d (first and second clause), it may enter into "search (for d) mode" and move to any other occurrence of d in the word. Then the automaton exits search mode (second last clause) and continues its computation either forward (last clause) or backward (see item 2).
5. $(s, 1) \in \delta_A(s, \ell)$, for each $s \in F$ and each $\ell \in \Sigma_A$. These transitions move the head of the automaton to the end of the input string, when the automaton enters a final state.

Observe that the separator symbol \$ does not allow transitions except in "search mode". Its role is to force the automaton to move in the correct direction when exiting "search mode".

The following theorem characterizes the relationship between an RPQI and the corresponding two-way automaton.

Theorem 3 *Let E be an RPQI, $\mathcal{K}$ a KB over $\mathcal{D}$, and $W_\mathcal{K}$ the word representing $\mathcal{K}$. Then A_E accepts $W_\mathcal{K}$ iff $ans(E, \mathcal{K})$ is nonempty.*

4 CHECKING CONTAINMENT

We characterize both the upper bound and the lower bound of the problem of checking containment between CRPQIs.

4.1 UPPER BOUND

Let Q_h, for $h = \{1, 2\}$, be in the form

$$Q_h(x_1, \ldots, x_n) \leftarrow y_{h,1}\, E_{h,1}\, y_{h,2} \wedge \cdots \wedge y_{h,2m_h-1}\, E_{h,m_h}\, y_{h,2m_h}$$

and let $\mathcal{V}_1, \mathcal{V}_2 \subseteq \Phi$ be the sets of variables of Q_1 and Q_2 respectively. To show that Q_1 is not contained in

Q_2, we have to search for a counterexample KB. From Theorem 2, we know that it is sufficient to look for a KB $\mathcal{K}$ and a mapping ν from the variables of Q_1 to the nodes of $\mathcal{K}$, such that $\mathcal{K}$ is ν-canonical for Q_1, and such that no $(Q_2, \mathcal{K}, \nu)$-mapping exists.

To generate candidate counterexample KBs and associated ν-mappings, we first construct a one-way automaton A_1 that accepts the set of words of the form

$$\$d_1 w_1 d_2 \$ d_3 w_2 d_4 \$ \cdots \$ d_{2m_1-1} w_{m_1} d_{2m_1} \$$$

that represent a KB that is ν-canonical for Q_1 for some ν, where each d_i is an element of a fixed set of nodes $\mathcal{D}_0$. We take $\mathcal{D}_0$ to be $2^{\mathcal{V}_1}$ and we explicitly encode in a word accepted by A_1 the mapping ν from the variables of Q_1 to the nodes in $\mathcal{D}_0$. This requires to ensure that the elements of $\mathcal{D}_0$ appearing in a word accepted by A_1 constitute a partition of $\mathcal{V}_1$ into equivalence classes. To construct A_1, we define:

- A one-way automaton $A_1^{Q_1}$ that accepts all words of the form above, where for $i \in \{1, \ldots, m_1\}$, $y_{1,2i-1} \in d_{2i-1}$, $y_{1,2i} \in d_{2i}$, and w_i is a word in $L(E_i)$.
- A one-way automaton A_1^p that checks that in a word w the distinct symbols $d_i \in \mathcal{D}_0$ appearing in w constitute a partition of $\mathcal{V}_1$.
- A one-way automaton A_1^a that checks that in a word w, whenever two symbols d_{2i-1} and d_{2i} are adjacent, then $d_{2i-1} = d_{2i}$.

The number of states in $A_1^{Q_1}$ is polynomial in the size of Q_1 (although the alphabet is exponential), while the number of states in A_1^p and A_1^a is exponential in the number of variables in Q_1. A_1 is the product automaton of $A_1^{Q_1}$, A_1^p, and A_1^a, and hence is of exponential size in Q_1. We call a word W accepted by A_1 a *Q_1-word*, and use $\mathcal{K}_W$ and ν_W to denote the KB and mapping corresponding to Q_1.

By virtue of the correspondence between Q_1-words and KBs that are ν-canonical for Q_1, our method for checking whether $Q_1 \not\subseteq Q_2$ is based on searching for a Q_1-word W such that no $(Q_2, \mathcal{K}_W, \nu_W)$-mapping exists. Now, let W be a Q_1-word. To check whether there is no $(Q_2, \mathcal{K}_W, \nu_W)$-mapping, we define a two-way automaton A_3 that checks the existence of such a mapping, and then complement A_3, obtaining an automaton A_4.

In order to define A_3, we represent $(Q_2, \mathcal{K}_W, \nu_W)$-mappings as *annotations* of Q_1-words, which specify where the variables of Q_2 are being mapped to in W, and hence in $\mathcal{K}_W$. More precisely, the Q_1-word $\$\ell_1 \cdots \ell_r\$$ with each symbol $\ell_i \neq \$$ annotated with γ_i is represented by the word $\$(\ell_1, \gamma_1) \cdots (\ell_r, \gamma_r)\$$ over the alphabet $((\Sigma \cup \mathcal{D}_0) \times 2^{\mathcal{V}_2}) \cup \{\$\}$. The intended meaning is that the variables in γ_i are mapped in $\mathcal{K}_W$ to the node ℓ_i, if $\ell_i \in \mathcal{D}_0$, and are mapped to the target node of the edge corresponding to the occurrence of ℓ_i, if $\ell_i \in \Sigma$.

Given a word W' representing an annotated Q_1-word W, an automaton A_2 can check if the annotation corresponds to a $(Q_2, \mathcal{K}_W, \nu_W)$-mapping. To construct A_2, we define:

1. A one-way automaton A_2^d that checks that for every symbol $d \in \mathcal{D}_0$ containing a distinguished variable x of Q_1, every occurrence of d in W is annotated in W' with a set of variables containing x.
2. A one-way automaton A_2^s that checks that for every variable $y \in \mathcal{V}_2$, either y appears in the annotation of at most one symbol in Σ, or it appears in the annotation of every occurrence of a symbol $d \in \mathcal{D}_0$.
3. A one-way automaton $A_2^\$$ that checks that every occurrence in W of a symbol preceding a \$ is annotated in W' with the same set of variables as the symbol preceding it.
4. A two-way automaton $A_2^{Q_2}$ that checks that for all $i \in \{1, \ldots, m_2\}$, the atom $y_{2,2i-1} E_{2,i} y_{2,2i}$ of Q_2 is satisfied in $\mathcal{K}_W$, i.e., there are symbols ℓ_1, ℓ_2 in W annotated in W' with γ_1 and γ_2 respectively, such that $y_{2,2i-1} \in \gamma_1$, $y_{2,2i} \in \gamma_2$, and the pair of nodes corresponding to ℓ_1 and ℓ_2 is in $ans(E_{2,i}, \mathcal{K}_W)$. To build $A_2^{Q_2}$ we exploit the construction in Section 3.

The number of states in $A_2^{Q_2}$ is polynomial in the size of Q_2, while the number of states in A_2^d, A_2^s, and $A_2^\$$ is exponential in the number of variables in Q_2. A_2 is the product automaton of A_2^d, A_2^s, $A_2^\$$, and of the one-way automaton equivalent to $A_2^{Q_2}$ [1]. Hence A_2 is of exponential size in Q_2.

Next we define the one-way automaton A_3 that simulates the guess of an annotation of a Q_1-word, and emulates the behaviour of A_2 on the resulting annotated word. The simulation of the guess and the emulation of A_2 can be obtained simply by constructing A_2 and then projecting out the annotation from the

[1] Notice that the number of states of the one-way automaton equivalent to $A_2^{Q_2}$ does not depend on the size of the alphabet, which is exponential in the number of variables of Q_1.

transitions. The idea behind this construction is that a path in A_3 from an initial state to a final state that leads to the acceptance of a non-annotated word W, corresponds to a path in A_2 that leads to the acceptance of a word W' which represents W with some annotation, and vice-versa. Observe that A_3 has the same number of states as A_2.

Let us stress that projecting out the annotations from the transitions corresponds to guess them. However, in order for the guesses to be meaningful the automaton must be one-way, since with a two-way automaton we cannot ensure that we make the same guess each time we pass over the same position in a word.

Finally, we define the one-way automaton A_4 as the complement of A_3.

Theorem 4 *Let Q_1 and Q_2 be two CRPQIs and A_1 and A_4 be as specified above. Then $Q_1 \not\subseteq Q_2$ iff $A_1 \cap A_4$ is nonempty.*

Proof (sketch). By Theorem 2, to check $Q_1 \not\subseteq Q_2$, it is sufficient to find a KB $\mathcal{K}$ and a mapping ν from the variables of Q_1 to the nodes of $\mathcal{K}$, such that $\mathcal{K}$ is ν-canonical for Q_1, and such that no $(Q_2, \mathcal{K}, \nu)$-mapping exists. Each word W accepted by A_1 represents a KB $\mathcal{K}_W$ and a mapping ν_W such that $\mathcal{K}_W$ is ν_W-canonical for Q_1. A_4 accepts a Q_1-word W iff there is no annotation of W that represents a $(Q_2, \mathcal{K}_W, \nu_W)$-mapping. Thus, $A_1 \cap A_4$ is nonempty iff there exists a Q_1-word W such that $\mathcal{K}_W$ is canonical for Q_1, and is such that no $(Q_2, \mathcal{K}_W, \nu_W)$-mapping exists. Hence, $A_1 \cap A_4$ is nonempty iff Q_1 is not contained in Q_2. □

Theorem 5 *Given two CRPQIs Q_1 and Q_2, checking whether $Q_1 \subseteq Q_2$ can be done in EXPSPACE.*

Proof. By theorem 4, $Q_1 \subseteq Q_2$ iff $A_1 \cap A_4$ is empty. The size of A_1 is exponential in the size of Q_1, the size of A_2 is exponential in the size of Q_2, the size of A_3 is polynomial in the size of A_2, and finally, the size of A_4 is exponential in the size of A_3. Therefore, the size of A_4 is doubly exponential in the size of Q_2. However, to check whether $A_1 \cap A_4$ is empty we do not need to construct A_4 explicitly. Instead, starting from A_3, we construct A_4 "on-the-fly"; whenever the emptiness algorithm wants to move from a state s_1 of the intersection of A_1 and A_4 to a state s_2, it guesses s_2 and checks that it is directly connected to s_1. Once this has been verified, the algorithm can discard s_1. Thus, at each step the algorithm needs to keep in memory at most two states and there is no need to generate all of A_4 at any single step of the algorithm. □

4.2 LOWER BOUND

Next we show an EXPSPACE lower bound for containment of conjunctive regular path queries without inverse (CRPQs). This closes the open problem in (Florescu et al., 1998b) on whether containment of CRPQs could be done in PSPACE.

To prove the result we exploit a reduction from tiling problems (van Emde Boas, 1982, 1997; Berger, 1966). A *tile* is a unit square of one of several types and the *tiling problem* we consider is specified by means of a finite set Δ of tile types, two binary relations H and V over Δ, representing horizontal and vertical adjacency relations, respectively, and two distinguished tile types $t_S, t_F \in \Delta$. The tiling problem consists in determining whether, for a given number n in unary, a region of the integer plane of size $2^n \times k$, for some k, can be tiled consistently with the adjacency relations H and K, and with the left bottom tile of the region of type t_S and the right upper tile of type t_F. Using a reduction from acceptance of EXPSPACE Turing machines analogous to the one in (van Emde Boas, 1997), it can be shown that this tiling problem is EXPSPACE-complete.

Theorem 6 *The problem of checking whether $Q_1 \subseteq Q_2$, where Q_1 and Q_2 are two CRPQs, is EXPSPACE-hard.*

Proof (sketch). Let $T = (\Delta, H, V, t_S, t_F)$ be an instance of the EXPSPACE-complete tiling problem above and n a number in unary. The alphabet is $\Sigma = \Delta \cup \{0, 1\}$. The query Q_1 is

$$Q_1(x_1, x_2) \leftarrow x_1\, E\, x_2$$

where the regular expression in the right hand side is

$$E = 0^n \cdot t_S \cdot ((0+1)^n \cdot \Delta)^* \cdot 1^n \cdot t_F.$$

Thus, a word in $L(E)$ consists of a sequence of *blocks*, each block consists of an n-bit *address* and a tile in Δ. We intend the sequence of addresses to behave as an n-bit counter, starting with 0^n and ending with 1^n. A word in $L(E)$ encodes a tiling if each pair of adjacent tiles is consistent with H and each pair of tiles that have the same address, where there is no tile in between them with the same address, is consistent with V. Thus, a word in $L(E)$ does not encode a tiling, if it contains an *error*. An error is a pair of blocks that exhibit an incorrect behavior of the counter or that has a pair of tiles that is inconsistent with H or V. We detect errors using the query Q_2, which is

$$Q_2(x_1, x_2) \leftarrow x_1\, E_1\, y_1 \wedge \Big(\bigwedge_{i \in \{0,\ldots,n\}} y_1\, F_i\, y_2 \Big) \wedge y_2\, E_1\, x_2$$

where E_1 is the regular expression $((0+1)^n \cdot \Delta)^*$, and F_i, for $i \in \{0, \ldots, n\}$ are constructed as explained below. The intuition is that y_1 and y_2 map to a pair that represents an error.

We define a regular expression F_C that detects adjacent blocks with an error in the address bits. F_C can be constructed by encoding an n-bit counter (Börger, Gräedel, & Gurevich, 1997).

We define a regular expression F_H that detects adjacent blocks in which the tiles do not respect the horizontal adjacency relation H:

$$F_H = \sum_{(t_1,t_2) \notin H} ((0+1)^n \cdot \Delta)^* \cdot (0+1)^n \cdot t_1 \cdot (0+1)^n \cdot t_2 \cdot ((0+1)^n \cdot \Delta)^*$$

In order to construct a regular expression that detects a sequence of 2^n+1 blocks, in which the tiles in the first and last block do not respect the vertical adjacency relation V, we define:

$$G_0 = \sum_{(t_1,t_2) \notin V} (0+1)^n \cdot t_1 \cdot ((0+1)^n \cdot \Delta)^* \cdot (0+1)^n \cdot t_2$$

and, for $i \in \{1, \ldots, n\}$, we define $G_i = G_i^0 + G_i^1$, where for $b \in \{0,1\}$ ($\overline{b}$ denotes the complement of bit b):

$$G_i^b = (0+1)^{i-1} \cdot b \cdot (0+1)^{n-i} \cdot \Delta \cdot (0+1)^* \cdot b \cdot (0+1)^* \cdot \Delta \cdot \overline{b}^n \cdot \Delta \cdot (0+1)^* \cdot b \cdot (0+1)^* \cdot \Delta \cdot (0+1)^{i-1} \cdot b \cdot (0+1)^{n-i} \cdot \Delta$$

Assuming that the address bits are correct, a word accepted by all $G_1, \ldots, G_n$ is constituted by a sequence of blocks in which the first and the last block are exactly 2^n blocks apart. This follows from the fact that the first and the last block coincide in all the address bits, and in between there is either exactly one block with all address bits equal to 0, or exactly one block with all address bits equal to 1. The intersection of $G_0, G_1, \ldots, G_n$ detects errors due to the vertical adjacency relation V.

Hence, by defining $F_i = G_i + F_H + F_C$, for $i \in \{0, \ldots, n\}$, the query Q_2 detects all three types of errors.

If there is no tiling, then every word in $L(E)$ contains an error and Q_1 is contained in Q_2. If there is a tiling, then there is a word in $L(E)$ without an error, and this word provides a counterexample to the containment of Q_1 in Q_2. □

5 CONCLUSIONS

We have presented both upper bound and lower bound results for containment of conjunctive regular path queries with inverse. This class of queries has several features that are typical of modern query languages for knowledge and data bases. In particular, it is the largest subset of query languages for XML data (Deutsch, Fernandez, Florescu, Levy, Maier, & Suciu, 1999) for which containment has been shown decidable.

The upper bound shows that adding inverse to conjunctive regular path queries does not increase the complexity of query containment. The lower bound holds also for the case without the inverse operator, and provides the answer to the question of which is the inherent complexity of checking containment of conjunctive regular path queries.

One interesting feature of our method is to demonstrate the power of two-way automata in reasoning on complex queries. The method can also be adapted to more general forms of queries and reasoning tasks. Indeed, it is easy to extend our algorithm to the case of union of conjunctive regular path queries with inverse.

Query containment is typically the first step in addressing the more involved problems of query rewriting and query answering using views. For the case of regular path queries, such problems have been studied in (Calvanese et al., 1999, 2000). We are currently working on extending these results to more powerful query languages, such as the one considered in this paper, by exploiting the techniques based on two-way automata.

Acknowledgments

This work was supported in part by the NSF grant CCR-970006, by MURST, by ESPRIT LTR Project No. 22469 DWQ (Foundations of Data Warehouse Quality), and by the Italian Space Agency (ASI) under project "Integrazione ed Accesso a Basi di Dati Eterogenee".

References

Abiteboul, S., Buneman, P., & Suciu, D. (1999). *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, Los Altos.

Adali, S., Candan, K. S., Papakonstantinou, Y., & Subrahmanian, V. S. (1996). Query caching and optimization in distributed mediator systems. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pp. 137–148.

Aho, A. V., Sagiv, Y., & Ullman, J. D. (1979). Equivalence among relational expressions. *SIAM J. on Computing, 8*, 218–246.

Berger, R. (1966). The undecidability of the dominoe problem. *Mem. Amer. Math. Soc., 66*, 1–72.

Börger, E., Gräedel, E., & Gurevich, Y. (1997). *The Classical Decision Problem.* Perspectives in Mathematical Logic. Springer-Verlag.

Bray, T., Paoli, J., & Sperberg-McQueen, C. M. (1998). Extensible Markup Language (XML) 1.0 – W3C recommendation. Tech. rep., World Wide Web Consortium. Available at http://www.w3.org/TR/1998/REC-xml-19980210.

Buneman, P. (1997). Semistructured data. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'97)*, pp. 117–121.

Buneman, P., Davidson, S., Hillebrand, G., & Suciu, D. (1996). A query language and optimization technique for unstructured data. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pp. 505–516.

Calvanese, D., De Giacomo, G., & Lenzerini, M. (1998). On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98)*, pp. 149–158.

Calvanese, D., De Giacomo, G., & Lenzerini, M. (1999). Representing and reasoning on XML documents: A description logic approach. *J. of Logic and Computation, 9*(3), 295–318.

Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., & Rosati, R. (1998). Description logic framework for information integration. In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'98)*, pp. 2–13.

Calvanese, D., De Giacomo, G., Lenzerini, M., & Vardi, M. Y. (1999). Rewriting of regular expressions and regular path queries. In *Proc. of the 18th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'99)*, pp. 194–204.

Calvanese, D., De Giacomo, G., Lenzerini, M., & Vardi, M. Y. (2000). Answering regular path queries using views. In *Proc. of the 16th IEEE Int. Conf. on Data Engineering (ICDE 2000)*. To appear.

Chan, E. P. F. (1992). Containment and minimization of positive conjunctive queries in oodb's. In *Proc. of the 11th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'92)*, pp. 202–211.

Chandra, A. K. & Merlin, P. M. (1977). Optimal implementation of conjunctive queries in relational data bases. In *Proc. of the 9th ACM Sym. on Theory of Computing (STOC'77)*, pp. 77–90.

Chaudhuri, S., Krishnamurthy, S., Potarnianos, S., & Shim, K. (1995). Optimizing queries with materialized views. In *Proc. of the 11th IEEE Int. Conf. on Data Engineering (ICDE'95)* Taipei (Taiwan).

Chaudhuri, S. & Vardi, M. Y. (1992). On the equivalence of recursive and nonrecursive Datalog programs. In *Proc. of the 11th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'92)*, pp. 55–66.

David S. Johnson, A. C. K. (1984). Testing containment of conjunctive queries under functional and inclusion dependencies. *J. of Computer and System Sciences, 28*(1), 167–189.

Deutsch, A., Fernandez, M., Florescu, D., Levy, A., Maier, D., & Suciu, D. (1999). Querying XML data. *IEEE Bulletin of the Technical Committee on Data Engineering, 22*(3), 10–18.

Donini, F. M., Lenzerini, M., Nardi, D., & Schaerf, A. (1996). Reasoning in description logics. In Brewka, G. (Ed.), *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pp. 193–238. CSLI Publications.

Donini, F. M., Lenzerini, M., Nardi, D., & Schaerf, A. (1998). $\mathcal{AL}$-log: Integrating Datalog and description logics. *J. of Intelligent Information Systems, 10*(3), 227–252.

Eklund, P., Nagle, T., Nagle, J., & Gerholz, L. (Eds.). (1992). *Conceptual Structures: Current Research and Practice.* Ellis Horwood.

Fensel, D., Knoblock, C., Kushmerick, N., & Rousset, M.-C. (Eds.). (1999). *IJCAI-99 Workshop on Intelligent Information Integration.* CEUR Electronic Workshop Proceedings, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-23/.

Fernandez, M., Florescu, D., Levy, A., & Suciu, D. (1999). Verifying integrity constraints on web-sites. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99).*

Fernandez, M. F. & Suciu, D. (1998). Optimizing regular path expressions using graph schemas. In *Proc. of the 14th IEEE Int. Conf. on Data Engineering (ICDE'98)*, pp. 14–23.

Florescu, D., Levy, A., & Mendelzon, A. (1998a). Database techniques for the World-Wide Web: A survey. *SIGMOD Record*, *27*(3), 59–74.

Florescu, D., Levy, A., & Suciu, D. (1998b). Query containment for conjunctive queries with regular expressions. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98)*, pp. 139–148.

Friedman, M., Levy, A., & Millstein, T. (1999). Navigational plans for data integration. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI'99)*. AAAI Press/The MIT Press.

Gupta, A. & Ullman, J. D. (1992). Generalizing conjunctive query containment for view maintenance and integrity constraint verification (abstract). In *Workshop on Deductive Databases (In conjunction with JICSLP)*, p. 195 Washington D.C. (USA).

Hopcroft, J. E. & Ullman, J. D. (1979). *Introduction to Automata Theory, Languages, and Computation.* Addison Wesley Publ. Co., Reading, Massachussetts.

Klug, A. C. (1988). On conjunctive queries containing inequalities. *J. of the ACM*, *35*(1), 146–160.

Knoblock, C. & Levy, A. Y. (Eds.). (1995). *Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Environments*, No. SS-95-08 in AAAI Spring Symposium Series. AAAI Press/The MIT Press.

Levy, A. Y., Rajaraman, A., & Ordille, J. J. (1996). Query answering algorithms for information agents. In *Proc. of the 13th Nat. Conf. on Artificial Intelligence (AAAI'96)*, pp. 40–47.

Levy, A. Y. & Rousset, M.-C. (1996). CARIN: A representation language combining Horn rules and description logics. In *Proc. of the 12th European Conf. on Artificial Intelligence (ECAI'96)*, pp. 323–327.

Levy, A. Y. & Rousset, M.-C. (1997). Verification of knowledge bases: a unifying logical view. In *Proc. of the 4th European Symposium on the Validation and Verification of Knowledge Based Systems* Leuven, Belgium.

Levy, A. Y. & Rousset, M.-C. (1998a). Combining horn rules and description logics in CARIN. *Artificial Intelligence*, *104(1-2)*, 165–209.

Levy, A. Y. & Rousset, M.-C. (1998b). Verification of knowledge bases based on containment checking. *Artificial Intelligence*, *101(1-2)*, 227–250.

Levy, A. Y. & Sagiv, Y. (1995). Semantic query optimization in datalog programs. In *Proc. of the 14th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'95)*, pp. 163–173.

Levy, A. Y. & Suciu, D. (1997). Deciding containment for queries with complex objects. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'97)*, pp. 20–31.

Milo, T. & Suciu, D. (1999). Index structures for path expressions. In *Proc. of the 7th Int. Conf. on Database Theory (ICDT'99)*, Vol. 1540 of *Lecture Notes in Computer Science*, pp. 277–295. Springer-Verlag.

Motro, A. (1996). Panorama: A database system that annotates its answers to queries with their properties. *J. of Intelligent Information Systems*, *7*(1).

Sagiv, Y. & Yannakakis, M. (1980). Equivalences among relational expressions with the union and difference operators. *J. of the ACM*, *27*(4), 633–655.

van der Meyden, R. (1992). *The Complexity of Querying Indefinite Information.* Ph.D. thesis, Rutgers University.

van Emde Boas, P. (1982). Dominoes are forever. In *Proc. of 1st GTI Workshop*, Rheie Theoretische Informatik UGH Paderborn, pp. 75–95 Paderborn (Germany).

van Emde Boas, P. (1997). The convenience of tilings. In Sorbi, A. (Ed.), *Complexity, Logic, and Recursion Theory*, Vol. 187 of *Lecture notes in pure and applied mathematics*, pp. 331–363. Marcel Dekker Inc.

Vardi, M. Y. (1989). A note on the reduction of two-way automata to one-way automata. *Information Processing Letters*, *30*(5), 261–264.

Widom, J. (1995). Special issue on materialized views and data warehousing. *IEEE Bulletin on Data Engineering*, *18*(2).

Reduction Rules and Universal Variables for First Order Tableaux and DPLL

Fabio Massacci
Dipartimento di Ingegneria dell'Informazione
Università di Siena
via Roma 56, 53100 Siena, Italy,
e-mail: massacci@dii.unisi.it

Abstract

Recent experimental results have shown that the strength of resolution, the propositional DPLL procedure, the KSAT procedure for description logics, or related tableau-like implementations such as DLP, is due to reduction rules which propagate constraints and prune the search space.

Yet, for first order logic such reduction rules are only known for resolution. The lack of reduction rules for first order tableau calculi (DPLL can be seen as a tableau calculus with semantic branching) is one of the causes behind the lack of efficient first order DPLL-like procedures.

The difficulty is that first order splitting rules force tableau and DPLL calculi to use rigid variables which hinder reduction rules such as unit subsumption or unit propagation.

This paper shows how reduction and simplification rules can be lifted to a first order tableau-like calculus using universal variables and a new rule called renaming. It also shows how to optimize the calculus for exploiting universal variables and reduction rules when semantic branching aka split DPLL-style is chosen as the inference rule with DFS-search.

1 Introduction

The last years have shown a clear trend in the development of efficient and effective procedures for automated reasoning. The renewed interest for experimental analysis and competition between systems has shown that what really makes a practical difference between theorem provers or satisfiability checkers is their ability of pruning the search space.

The importance of rules for reducing the search space has been stressed since the very beginning in the design of the Davis-Putnam-Loveland-Procedure [11, 10] and in the early Russian studies in automated deduction [28]. The ability of performing fast and effective unit propagation has been a key for the successes in finite algebra of the theorem prover SATO by Zhang [33].

In the realms of modal and description logics, a major qualitative advance has been the introduction of the same pruning techniques of DPLL in the KSAT decision procedure for modal logic K by Giunchiglia and Sebastiani [16]. Further experimental analysis by Hustadt and Schmidt [20] has confirmed the importance of reduction rules over inference rules: once reduction rules were added to the tableau system KRIS its performance increased by orders of magnitude. The DLP system by Patel-Schneider, "winner" of the recent TANCS competition of modal logics [24], uses many reduction rules such as boolean constraint propagation [19].

In first order logic, the importance of reduction rules for resolution (such as subsumption) has been stressed since its inception by Robinson [27] and its implementation has been the subject of an intense study (see e.g. [14, 30]). The competitive results by Hustadt and Schmidt [20] for the translation of modal logic into first order logic are also due to the subsumption routines of the prover SPASS. The performance of Vampire at the last CASC competition is mainly due to its sophisticated subsumption architecture by Voronkov [30]

If we were allowed to condense these experimental findings in one sentence we could just say that *reduction rules are more effective than inference rules.* In practice, the most (if not the only) effective reduction rules are those which act locally, affect one or few formulae at the time and do not require to compare whole parts of the proof search tree. The locality of the rules makes then possible optimized implementations in which such rules are applied to all formulae at once [30, 33].

The practical importance of local reduction rules also explains one surprising duality in experimental results: in the propositional and modal realms, tableau-like systems (DPLL can be seen as a tableau calculus using semantic branching) are the undisputed front runners; in the first-order domain, systems based on hyper-resolution or superposition (e.g. Vampire, Gandalf, SPASS) take the front stage.

Indeed, for propositional, modal, and description logics, local, formula-wise reduction rules are well studied and go along with different forms of tableau calculi (i.e. branching rules) and can fully exploit the features of the underlying logic [16, 18, 19, 23].

In contrast, for first order logic, a key formalisms for knowledge representation, there is only one competitor which fully exploits first order features: the subsumption rule for resolution (and superposition). Techniques for search pruning in tableau-based approach such as factoring and merging do exist but they require considering whole parts of the proof tree [1, 21, 32]. In other cases, such as the first order version of the DPLL procedure [11], reduction rules are gained only by grounding formulae to propositional logic.

The major problem is that first order branching rules force tableau, model elimination, and DPLL-like calculi to use rigid variables[1] and this substantially hinders the ability of performing first order local reduction rules such as unit subsumption or unit propagation.

This paper proposes a general solution based on the observation that in the resolution framework all variables are "implicitly quantified". We propose a modified tableau calculus in which all variables are also implicitly universally quantified unless we explicitly mark them as rigid. Then, we introduce a rule called *renaming* which generalizes the use of universal variables proposed by Beckert et al. [6, 7] and by Baumgarten [4], and the use of conjunctive superformulae proposed by Degtyarev and Voronkov [13].

In this new framework we can now introduce full first order reduction rules for tableau-like calculi and prove them sound (and sometimes invertible). We also show how the calculus can be optimized to fully exploit universal variables and reduction rules when semantic branching aka split DPLLstyle is chosen as the main inference rule with a depth-first-search strategy and discuss how a lean implementation in sicstus prolog can be done for this calculus.

The introduction of first order reduction rules provides also an *unifying perspective* of many (tableau based) deduction techniques and "explains away" the stand-alone (inefficient) nature of the classical first order DPLL.

2 Notation and Terminology

We assume a basic knowledge of first order syntax and semantics [15, 29]. First order *terms*, denoted by t, are constructed from constants $a, b, c \in \mathcal{C}on$ and variables $x, y, z \in \mathcal{V}ar$ using functions $f, g \in \mathcal{F}un$. The usual definition of ground terms (terms without variables) is adopted.

We construct *formulae* A, B, C from predicate symbols $P(t_1, \ldots, t_n)$ and $Q(t_1, \ldots t_n)$ and the boolean constants $\top$ and $\bot$ with the propositional connectives $\wedge$, $\vee$, $\neg$, and the first order connectives $\forall$ and $\exists$: e.g. $A \wedge B, \forall x.A, \exists x.A$. Other connectives can be seen as abbreviations, e.g. $A \supset B \doteq \neg A \vee B$.

If A is a formula, its *conjugate* $\overline{A}$ is obtained in the standard way: $\overline{\neg A} = A$ and $\overline{A} = \neg A$ if the main connective of A is not a negation.

By $FV(A)$ we denote the *set of free variables occurring in the formula* A. By $A(x)$ we mean that the variable x may occur free in the formula A and $A(t)$ denotes the simultaneous substitution of t in all (if any) occurrences of x in A. $FV(\cdot)$is extended to set of formulae or set of terms in the obvious way. A *sentence* is a formula A such that $FV(A) = \emptyset$.

By $\{t_1/x_1, \ldots, t_n/x_n\}$ we denote the *substitution* which simultaneously replaces all occurrences of each variable x_i with the corresponding term t_i. Substitutions are usually denoted by σ. A *renaming*, denoted by η, is a particular substitution $\{y_1/x_1, \ldots, y_n/x_n\}$ where all y_i are fresh variables.

By $A\sigma$ we denote result of the application of the substitution σ to the formula A. We employ the usual concept of *substitution free for a formula* A, i.e. the variables of t_i do not get bound after the substitution of t_i for x_i. To this extent we rename, if necessary, bound variables in a formula before applying a substitution. In the sequel substitutions are assumed free.

The *domain* of a substitution $\{t_1/x_1, \ldots, t_n/x_n\}$ is the set of variables $\{x_1, \ldots, x_n\}$. If $\mathcal{R}$ is a set of variables we denote by $\sigma_{\overline{\mathcal{R}}}$the restriction of the domain of σ to variables which are *not* in $\mathcal{R}$, i.e. $domain(\sigma_{\overline{\mathcal{R}}}) \cap \mathcal{R} = \emptyset$.

[1] Loosely speaking, when we split the search in two branches with $P(x)$ on one side and and $\neg P(x)$ on the other, the variable x in $P(x)$ is called rigid because choosing its value affects another branch. As a consequence, $P(x)$ cannot be used to unit-subsume $P(b) \vee Q$, because we are not free to let x range over all possible values.

3 A First Order Calculus with Renaming

Before introducing first order reduction rules, we need a calculus which can keep track of rigid and universal (i.e. implicitly universally quantified) variables. Standard free-variables tableau calculi use only rigid variables [26, 15, 13, 31]; more advanced methods make a limited use of universal variables [6, 7, 5, 4]. Here all variables are assumed to be universal ones unless we explicitly consider them rigid.

Definition 1 *A* prefixed formula *is a pair $\mathcal{R}{:}A$ where A is a first order formula and $\mathcal{R}$ is a set of variables. A* prefixed sequent, *denoted by $\mathcal{S}$, is a set of prefixed formulae.*

The variables in $\mathcal{R}$ are the only *rigid variables* whose value may affect other parts of the proof search tree. All free variables in A which are not in $\mathcal{R}$ are implicitly universally quantified. In the sequel we use $\mathcal{S}, \mathcal{R}{:}A$ as a short-cut for $\mathcal{S} \cup \{\mathcal{R}{:}A\}$ and $\mathcal{R}, x$ as a shortcut for $\mathcal{R} \cup \{x\}$.

The definition of search tree is standard [15, 29]:

Definition 2 *A* proof search tree $\mathcal{T}$ *for a formula A is a dyadic tree where each node is labelled by a prefixed sequent so that*

- *the root is labelled by the prefixed sequent $\{FV(A) : \neg A\}$;*
- *if a node is labelled by a sequent its children are labelled with the corresponding consequents of a rule of the calculus from Fig. 1 and Fig. 2.*

Intuitively, we may say that a child of a node is actually labelled by replacing one formula in the sequent of its parents by the consequence of a rule applied to that formula, and then copying the other formulas in the sequent unchanged.

Remark. For what regard visual presentation, the *proof search tree grows upward*, with the sentence to be proved at the bottom and the leaves (axioms) at the top.

In the sequel we use the notion of a *frontier* of a proof search tree $\mathcal{T}$: it is the collection of all leaves of the tree and is denoted by $\mathcal{S}_1 \mid \ldots \mid \mathcal{S}_n$. If we view the proof search tree as a tableau, the frontier is simply the collection of the current branches of the tableau [7, 13, 15].

Figure 1 contains rules which do not introduce new branches in the proof search tree. A general observation is that we assume formulae to be "definitively"

$$\frac{\mathcal{S}, \mathcal{R}{:}A, \mathcal{R}{:}B}{\mathcal{S}, \mathcal{R}{:}A \wedge B}\ \alpha_\wedge \qquad \frac{\mathcal{S}, \mathcal{R}{:}\neg A, \mathcal{R}{:}\neg B}{\mathcal{S}, \mathcal{R}{:}\neg(A \vee B)}\ \alpha_\vee \qquad \frac{\mathcal{S}, \mathcal{R}{:}A}{\mathcal{S}, \mathcal{R}{:}\neg\neg A}\ \alpha_\neg$$

$$\frac{\mathcal{S}, \mathcal{R}{:}A(y)}{\mathcal{S}, \mathcal{R}{:}\forall x.A(x)}\ \gamma_\forall \qquad \frac{\mathcal{S}, \mathcal{R}{:}\neg A(y)}{\mathcal{S}, \mathcal{R}{:}\neg\exists x.A(x)}\ \gamma_\exists$$

$$\frac{\mathcal{S}, \mathcal{R}{:}\neg A(f_A(x_1 \ldots x_n))}{\mathcal{S}, \mathcal{R}{:}\neg\forall x.A(x)}\ \delta_\forall \qquad \frac{\mathcal{S}, \mathcal{R}{:}A(f_A(x_1 \ldots x_n))}{\mathcal{S}, \mathcal{R}{:}\exists x.A(x)}\ \delta_\exists$$

$$\frac{\mathcal{S}, \mathcal{R}{:}A\eta_{\overline{\mathcal{R}}}}{\mathcal{S}, \mathcal{R}{:}A}\ (ren) \qquad \frac{\mathcal{S}, \mathcal{R} \cap FV(A){:}A}{\mathcal{S}, \mathcal{R}{:}A}\ (norm)$$

Where y is a fresh variable in the γ-rules; f_A is a skolem function symbol and $\{x_1, \ldots x_n\}$ is the set of free variables of A in the δ-rules.

Figure 1: First order calculus with renaming

reduced after an application of a rule, unless they are not explicitly duplicated in the consequent(s). So we only have duplication in the β-rules of Fig. 2.

There are various ways to optimize the generation of skolem functions in the δ rules for the existential quantifiers. This topic is orthogonal to the one we are treating here, and we refer to the literature for discussion [17, 3].

The key rule is the (ren) rule, which we use to boost the usage of universal variables. Indeed its task is to minimize the number of variables that are shared between formulae. In this way, the substitution of a term for a variable in a formula will not affect other formulae, unless the variable is rigid. Notice that the renaming $\eta_{\overline{\mathcal{R}}}$ is *local* to the formula A: it is not applied to the whole sequent $\mathcal{S}$ nor, a fortiori, to the whole search tree. This renaming rule is such that the standard soundness proof of rigid and universal variables tableau calculi fails for it, so we will use a novel technique to prove its soundness.

In general, renaming and normalization should be immediately applied to the formulae newly introduced by other rules.

Figure 2 contains two alternative set of rules for branching. The first set is a variant of the classical β-rule of tableau calculi whereas the second set of rules is usually called semantic branching (see e.g. the modal optimizations by Horrocks and Patel-Schneider [19]).

Branching rules are responsible for the introduction of rigid variables. Indeed if we have $A(x) \vee B(x)$ and found out that $A(x)$ is contradictory for some substitution t/x, we have to propagate this substitution also to the branch of the search tree containing the disjunct $B(x)$. This operation is costly and, if the implementation does not guarantee proof-confluence as in [4]

Rules for symmetric branching:

$$\frac{\mathcal{S}, \mathcal{R}{:}A \vee B, \mathcal{R}_{A \bullet B}{:}A \quad \mathcal{S}, \mathcal{R}{:}A \vee B, \mathcal{R}_{A \bullet B}{:}B}{\mathcal{S}, \mathcal{R}{:}A \vee B}\ \beta_\vee \qquad \frac{\mathcal{S}, \mathcal{R}{:}\neg(A \wedge B), \mathcal{R}_{A \bullet B}{:}\neg A \quad \mathcal{S}, \mathcal{R}{:}\neg(A \wedge B), \mathcal{R}_{A \bullet B}{:}\neg B}{\mathcal{S}, \mathcal{R}{:}\neg(A \wedge B)}\ \beta_\wedge$$

Rules for semantic branching:

$$\frac{\mathcal{S}, \mathcal{R}{:}A \vee B, \mathcal{R}_A{:}A \quad \mathcal{S}, \mathcal{R}{:}A \vee B, \mathcal{R}_{A \bullet B}{:}B, \mathcal{R}_A{:}\neg A}{\mathcal{S}, \mathcal{R}{:}A \vee B}\ s\beta_\vee \qquad \frac{\mathcal{S}, \mathcal{R}{:}\neg(A \wedge B), \mathcal{R}_A{:}\neg A \quad \mathcal{S}, \mathcal{R}{:}\neg(A \wedge B), \mathcal{R}_{A \bullet B}{:}\neg B, \mathcal{R}_A{:}A}{\mathcal{S}, \mathcal{R}{:}\neg(A \wedge B)}\ s\beta_\wedge$$

where $\mathcal{R}_{A \bullet B} = \mathcal{R} \cup (FV(A) \cap FV(B))$ and $\mathcal{R}_A = \mathcal{R} \cup FV(A)$

Figure 2: Symmetric and semantic branching rules for the first order calculus with renaming

we might be forced to backtrack over previous unifications. Thus, a major task is to minimize the number of rigid variables introduced after a branching point.

Symmetric branching allows for a good minimization of rigid variables in both branches: the rigid variables introduced by the symmetric branching rule are only the free variables of A and B which occur in both conjuncts. We shall see that we can do better once a depth-first strategy is assumed. Unfortunately, semantic branching is not equally well suited. To retain soundness we are forced to keep rigid all variables of the first disjunct A. An alternative is to introduce new skolem terms in the $\neg A$ added on the right branch and we have not pursued it here.

The differences between branching rules can be intuitively explained by considering the formula $\forall xyz.P(x,y) \vee Q(y,z)$. This formula is logically equivalent to the following three formulae:

1. $\forall y.\,(\forall x.P(x,y)) \vee (\forall z.Q(y,z))$
2. $\forall yx.P(x,y) \vee (\neg P(x,y) \wedge \forall z.Q(y,z))$
3. $\forall y.(\forall x.P(x,y)) \vee ((\exists x.\neg P(x,y)) \wedge (\forall z.Q(y,z)))$

Loosely speaking, the first alternative is what our rule for symmetric branching does. The second alternative is what we have chosen for semantic branching and the third one is what we decided to discard, because the optimized version of the semantic branching rule makes it possible to simulate its effects without introducing new skolem terms (see section 5).

One may also wonder why we do not just pre-process the input following the intuition above: after all this is a variant of a technique known as *mini-scoping* which is used for skolemization by theorem provers like SPASS. The problem is that mini-scoping is static, whereas in with an explicit representation of rigid variables we can benefit from the universal variables that are introduced during the search.

We can now state what a proof is:

Definition 3 *A sequent $\mathcal{S}$ is closed by a substitution σ iff either $\mathcal{R}{:}\neg\top \in \mathcal{S}$, $\mathcal{R}{:}\bot \in \mathcal{S}$, or there are two formulae $\mathcal{R}_A{:}A \in \mathcal{S}$ and $\mathcal{R}_B{:}B \in \mathcal{S}$ such that $A\sigma = \overline{B\sigma}$.*

Definition 4 *A pair $\langle \mathcal{T}, \sigma_\mathcal{T} \rangle$, where $\mathcal{T}$ is a search tree and σ a substitution, is a* proof *for the sentence A if*

- *$\{\emptyset{:}\neg A\}$ labels the root of the tree $\mathcal{T}$;*
- *each $\mathcal{S}_i$ in the frontier of $\mathcal{T}$ is closed by $\sigma_\mathcal{T}$*

We refer to $\sigma_\mathcal{T}$ as a *closing substitution* for $\mathcal{T}$. We have chosen this formulation so that one can apply the substitution $\sigma_\mathcal{T}$ to the whole tableau and obtain a result close to a standard proof search tree of the ground version of tableaux [15] or DPLL [11].

In alternative, if we choose to represent only the frontier of the tree, as done by Beckert and Possega [7] or Degtyarev and Voronkov [13], we could have described the proof as a sequence of tableau and substitutions as in [7] or the sequent elimination rule (*abc*) as in [13]. For instance, the (*abc*) rule allows to reduce the frontier of not-closed sequents $\mathcal{S}_1 \mid \mathcal{S}_2 \mid \ldots \mid \mathcal{S}_n$ into the frontier $\mathcal{S}_2\sigma \mid \ldots \mid \mathcal{S}_n\sigma$ provided that σ is a closing substitution for $\mathcal{S}_1$.

Since the definition of proof is fairly close to the classical one, we may wonder where renaming plays a role. The key observation is that, if we apply renaming to all formulae of a leaf then the only shared variables between two leaves will be the rigid variable introduced at some branching points. After renaming we can use the same formula for closing different branches. For instance, one may consider the formula $(\forall x.P(x)) \wedge (\neg P(a) \vee P(b))$. We have two branches $\{\emptyset{:}\neg P(a), \emptyset{:}P(x)\}$ and $\{\emptyset{:}\neg P(b), \emptyset{:}P(x)\}$ and we can close them simultaneously rename x into x_1 in the first branch and into x_2 in the second one.

Renaming makes us available as many instances of non-rigid variables as we want. Thus, if new rigid variables are not introduced, an optimization is possible:

Proposition 1 *In the branching rules of Figure 2, the disjunction $A \vee B$ can be deleted from both consequents if it is a ground formula.*

The first result of this paper is then the following:

Theorem 1 *A sentence A is valid iff there is a proof for A.*

So far we have not tackled the issue of fairness or termination. The simplest (but least effective) technique is to use a bound on the number of times that a disjunction can be duplicated by the β-rules and then use backtracking over all possible unifications for a given bound. Mutatis mutandis, this strategy is employed by the standard prolog implementation of first order tableau calculi [15, 7].

Advanced implementations, such as the Hypertableaux calculus [5], use a bound on the depth of terms that are used by the unification $\sigma_{\mathcal{T}}$ for closing each branch. This technique can be lifted to this framework by introducing a notion of reduced formula (modulo renaming), using depth first search and forbidding substitutions that, when applied to the current sequent to close it, generate two instances of the same disjunction. We sketch its implementation in section 6.

4 First Order Reduction Rules

The next step is the definition of the reduction rules.

The basic operation is *boolean reduction.* This operation, sometimes called lexical normalization or boolean constraint propagation, is one of the strength of description logic theorem provers such as DLP [19], KSAT [16] or FaCT [18]. Advanced modal and propositional reductions have been also proposed by Massacci [23].

We denote the result of boolean reduction of a formula A by $A\downarrow$, and the corresponding rules are shown in Fig. 3. Although this reduction already yields a substantial simplification of a formula, it is still not sufficient as it does not fully exploit the power of first order logic.

For first order reductions we must take care of rigid variables. Thus, if $\mathcal{R}$ is a set of rigid variables, we denote the result of the *first order reduction modulo* $\mathcal{R}$ of a formula A as $(A)\Downarrow_{\mathcal{R}}$. Corresponding rules are in Fig. 4.

The main intuition is that we cannot use rigid variables for unification and therefore we must keep track of which variable is rigid and which is (implicitly) universal. This explains the treatment of the existentially quantified variables, which are added to the set of rigid variables as the reduction proceeds down to the subformulae, and the treatment of disjunction, in which we must add the set of variables shared between the two disjuncts to the set of rigid variables. Once rigid variables are set aside, we can exploit the fact that all other variables are implicitly universally quantified.

For example, in the replacement of two conjuncts by $\top$, suppose we have the formula $\forall xyz.(P(x,a) \wedge \neg P(f(y),y)) \vee Q(z)$. Clearly, the variables x, y are implicitly universally quantified. So, whenever this formula is introduced in the proof search tree we can apply a γ rule, a β and then an α rule. The resulting formulae $P(x,a)$ and $\neg P(f(y),y)$, thanks to renaming, can immediately close the sequent without affecting any other part of the search tree with the substitution $\{f(a)/x, a/y\}$. Thus, we can gain from this knowledge and immediately replace our subformula with $\bot$.

For the merging of two conjuncts into one formula, suppose that we have the formula $\forall xy.P(f(x)) \wedge P(y)$. Whenever this formula is added to the proof search tree we obtain $\emptyset : P(f(x))$ and $\emptyset : P(y)$. Now the second formula obviously subsumes the first. With renaming we can generate fresh instances of $P(y)$ in different sequent leaves of the proof search tree and we never need to use $P(f(x))$. Thus, we keep only the most general conjunct.

Disjunction is subtler. At first we have not used unification but only identity for the $\top$ clause of the disjunction. If we had used unification the calculus would have been incomplete. Consider the formula

$$(\neg P(a) \vee \neg P(b)) \wedge (P(c) \vee P(d)) \wedge (\forall x.\forall y.P(x) \vee \neg P(y)).$$

This formula is unsatisfiable but if we had applied first order reduction using unification we would have "reduced" it to a satisfiable formula.

Remark. The conditions for merging two disjuncts into one disjunct is the dual of the condition used for conjunction: we keep only the most specific disjunct. This is the only place where we must give away equivalence preserving rules in favor of satisfiability preserving rules.

Finally we can devise the last and most important reduction rule: *simplification.* The intuition is that once a subformula is added to a sequent, we commit to consider it true and then we may simplify other formulae using this information. We denote the simplification of a formula A using another prefixed formula $\mathcal{R}{:}B$ as $A[\mathcal{R} : B]$ and show its working in Fig. 5. The formula A is sometimes referred to as the simplified formula and the formula B as the simplifying formula.

Notice that the substitution $\sigma_{\overline{\mathcal{R}}}$ is not applied to the

$$
\begin{array}{rcl}
(A \wedge B)\downarrow & = & \begin{cases} \bot & \text{if } A\downarrow = \bot \text{ or } B\downarrow = \bot \text{ or } A\downarrow = \overline{B\downarrow} \\ A\downarrow & \text{if } A\downarrow = B\downarrow \\ A\downarrow \wedge B\downarrow & \text{otherwise} \end{cases} \\
(A \vee B)\downarrow & = & \begin{cases} \top & \text{if } A\downarrow = \top \text{ or } B\downarrow = \top \text{ or } A\downarrow = \overline{B\downarrow} \\ A\downarrow & \text{if } A\downarrow = B\downarrow \\ A\downarrow \vee B\downarrow & \text{otherwise} \end{cases} \\
\neg\neg A\downarrow & = & A\downarrow \\
\neg(A \wedge B)\downarrow & = & \begin{cases} \top & \text{if } \neg A\downarrow = \top \text{ or } \neg B\downarrow = \top \text{ or } \neg A\downarrow = \overline{\neg B\downarrow} \\ \neg A\downarrow & \text{if } \neg A\downarrow = \neg B\downarrow \\ \neg(A\downarrow \wedge B\downarrow) & \text{otherwise} \end{cases} \\
\neg(A \vee B)\downarrow & = & \begin{cases} \bot & \text{if } \neg A\downarrow = \bot \text{ or } \neg B\downarrow = \bot \text{ or } \neg A\downarrow = \overline{\neg B\downarrow} \\ \neg A\downarrow & \text{if } \neg A\downarrow = \neg B\downarrow \\ \neg(A\downarrow \vee B\downarrow) & \text{otherwise} \end{cases}
\end{array}
$$

Figure 3: Boolean reduction rules

$$
\begin{array}{rcl}
(\neg\neg A)\Downarrow_{\mathcal{R}} & = & (A)\Downarrow_{\mathcal{R}} \\
(\forall x.A)\Downarrow_{\mathcal{R}} & = & \begin{cases} (A)\Downarrow_{\mathcal{R}} & \text{if } x \notin FV((A)\Downarrow_{\mathcal{R}}) \\ \forall x.(A)\Downarrow_{\mathcal{R}} & \text{otherwise} \end{cases} \\
(\exists x.A)\Downarrow_{\mathcal{R}} & = & \begin{cases} (A)\Downarrow_{\mathcal{R},x} & \text{if } x \notin FV((A)\Downarrow_{\mathcal{R},x}) \\ \forall x.(A)\Downarrow_{\mathcal{R},x} & \text{otherwise} \end{cases} \\
(\neg\forall x.A)\Downarrow_{\mathcal{R}} & = & \begin{cases} (\neg A)\Downarrow_{\mathcal{R},x} & \text{if } x \notin FV((\neg A)\Downarrow_{\mathcal{R},x}) \\ \neg\forall x.\neg((\neg A)\Downarrow_{\mathcal{R},x})\downarrow & \text{otherwise} \end{cases} \\
(\neg\exists x.A)\Downarrow_{\mathcal{R}} & = & \begin{cases} (\neg A)\Downarrow_{\mathcal{R}} & \text{if } x \notin FV((\neg A)\Downarrow_{\mathcal{R}}) \\ \neg\exists x.\neg((\neg A)\Downarrow_{\mathcal{R},x})\downarrow & \text{otherwise} \end{cases} \\
((A \wedge B))\Downarrow_{\mathcal{R}} & = & \begin{cases} \bot & \text{if } (A)\Downarrow_{\mathcal{R}} = \bot \text{ or } (B)\Downarrow_{\mathcal{R}} = \bot \\ & \text{or } ((A)\Downarrow_{\mathcal{R}})\sigma_{\overline{\mathcal{R}}} = \overline{((B)\Downarrow_{\mathcal{R}})\sigma_{\overline{\mathcal{R}}}} \text{ for some substitution } \sigma \\ (A)\Downarrow_{\mathcal{R}} & \text{if } ((A)\Downarrow_{\mathcal{R}})\sigma_{\overline{\mathcal{R}}} = (B)\Downarrow_{\mathcal{R}} \\ (B)\Downarrow_{\mathcal{R}} & \text{if } (A)\Downarrow_{\mathcal{R}} = ((B)\Downarrow_{\mathcal{R}})\sigma_{\overline{\mathcal{R}}} \\ (A)\Downarrow_{\mathcal{R}} \wedge (B)\Downarrow_{\mathcal{R}} & \text{otherwise} \end{cases} \\
((A \vee B))\Downarrow_{\mathcal{R}} & = & \begin{cases} \top & \text{if } (A)\Downarrow_{\mathcal{R}_{A\bullet B}} = \top \text{ or } (B)\Downarrow_{\mathcal{R}_{A\bullet B}} = \top \\ & \text{or } (A)\Downarrow_{\mathcal{R}_{A\bullet B}} = \overline{(B)\Downarrow_{\mathcal{R}_{A\bullet B}}} \\ (A)\Downarrow_{\mathcal{R}_{A\bullet B}} & \text{if } (A)\Downarrow_{\mathcal{R}_{A\bullet B}} = ((B)\Downarrow_{\mathcal{R}_{A\bullet B}})\sigma_{\overline{\mathcal{R}_{A\bullet B}}} \\ (B)\Downarrow_{\mathcal{R}_{A\bullet B}} & \text{if } ((A)\Downarrow_{\mathcal{R}_{A\bullet B}})\sigma_{\overline{\mathcal{R}_{A\bullet B}}} = (B)\Downarrow_{\mathcal{R}_{A\bullet B}} \\ (A)\Downarrow_{\mathcal{R}_{A\bullet B}} \wedge (B)\Downarrow_{\mathcal{R}_{A\bullet B}} & \text{otherwise} \end{cases}
\end{array}
$$

Figure 4: First order reduction

$$
A[\mathcal{R}:B] = \begin{cases} \top & \text{if } A = B\sigma_{\overline{\mathcal{R}}} \text{ for some } \sigma \\ \bot & \text{if } \overline{A} = B\sigma_{\overline{\mathcal{R}}} \text{ for some } \sigma \\ \neg(C[\mathcal{R}:B]) & \text{if } A = \neg C \\ (C[\mathcal{R}:B]) \wedge (D[\mathcal{R}:B]) & \text{if } A = C \wedge D \\ (C[\mathcal{R}:B]) \vee (D[\mathcal{R}:B]) & \text{if } A = C \vee D \\ \forall x(C[\mathcal{R}:B]) & \text{if } A = \forall x C \\ \exists x(C[\mathcal{R}:B]) & \text{if } A = \exists x C \end{cases}
$$

Figure 5: First order simplification

$$\frac{\mathcal{S},\mathcal{R}{:}A\downarrow}{\mathcal{S},\mathcal{R}{:}A}\ bool-red \qquad \frac{\mathcal{S},\mathcal{R}{:}(A)\Downarrow_{\mathcal{R}}}{\mathcal{S},\mathcal{R}{:}A}\ fo-red$$

$$\frac{\mathcal{S},\mathcal{R}_A{:}A\,[\mathcal{R}_B:B],\mathcal{R}_B{:}B}{\mathcal{S},\mathcal{R}_A{:}A,\mathcal{R}_B{:}B}\ simp$$

Figure 6: First order reduction rules

tableau nor the simplified formula, nor it modifies the rigid variables in B. This is in common to many techniques used by clausal tableaux [5, 21]. We go further than that since different substitutions can be used for different occurrences of a subformula. So the substitution is local to the subformula.

For instance, if we apply simplification as follows

$$\forall y.((\neg P(x,a)\vee Q(y))\wedge(Q(b)\vee\neg P(x,b)))\,[\{x\}:P(x,y)]$$

we can use the substitution a/y for $P(x,a)$ and the substitution b/y for $P(x,b)$. Then we get the formula $\forall y.(\bot\vee Q(y))\wedge(Q(b)\vee\bot))$ and by first order reduction we get just $\forall y.Q(y)$.

The proper reduction rules for our first order calculus with renaming are shown in Fig. 6.

Then we can state the other main results of this paper:

Theorem 2 *The boolean reduction rule is a sound and invertible rule for the first order calculus with renaming.*

Theorem 3 *The first order reduction rule is a sound rule for the first order calculus with renaming.*

Theorem 4 *The first order simplification rule is a sound and invertible rule for the first order calculus with renaming.*

5 Optimizations for Branching Rules

The usage of universal variables in the branching rules presented in Fig. 2 can be further optimized if a depth first strategy is used. The optimization is also necessary for semantic branching since we have only a limited number of universal variables in one branch due to the presence of $\mathcal{R}_A{:}A$ on one branch and $\mathcal{R}_A{:}\neg A$ on the other of branch the search tree.

The intuition is that with a depth first strategy the amount of information that is available when the right branch of a disjunct is considered is much more than that actually expressed by the formulae introduced by the rule. Indeed we already know that the left branch labelled with $\mathcal{S},\mathcal{R}_A:A$ can be closed by a certain substitution σ_A. In many cases, the formula on the right $\mathcal{R}_A:\neg A\sigma_A$ will not be ground. With the current branching rule all remaining free variables will be treated rigidly. The key of the optimization is how can we transform (some of) them into universal variables.

We need a definition:

Definition 5 *Let $\mathcal{S} = \{\mathcal{R}_1{:}A_1,\ldots,\mathcal{R}_n{:}A_n\}$ be a sequent and σ be a substitution. The result of the application of σ to $\mathcal{S}$ is the sequent $\{FV(\mathcal{R}_1\sigma){:}A_1\sigma,\ldots,FV(\mathcal{R}_n\sigma){:}A_n\sigma\}$.*

It can be shown that if $\mathcal{R}_i$ contains the only free variables of A_i which are shared with some other A_j then one also has that $FV(\mathcal{R}_i\sigma)$ contains the only free variables of $A_i\sigma$ which are shared with another $A_j\sigma$.

Then we can optimize the symmetric branching rules of Fig. 2 as follows. Let $\mathcal{S},\mathcal{R}{:}A\vee B$ be a sequent and let σ_A be a closing substitution for the proof search tree whose root is labelled by $\mathcal{S},\mathcal{R}{:}A\vee B,\mathcal{R}_{A*B}{:}A$. Then the right successor of $\mathcal{S},\mathcal{R}{:}A\vee B$ in the proof search tree can be $\mathcal{S}\sigma_A,FV(\mathcal{R}\sigma_A){:}B\sigma_A$. Notice that we use $FV(\mathcal{R}\sigma_A)$ and not $FV(\mathcal{R}_{A*B}\sigma_A)$.

The optimization can be better understood if we reason about the transformation of the *active frontier* of the tree constituted by the leaves of the proof search tree which are not (yet) closed by our procedure for constructing the search tree in depth-first. Assume that the frontier is the following:

$$\mathcal{S}_A,\mathcal{R}{:}A\vee B,\mathcal{R}_{A*B}{:}A\mid\mathcal{S}_B,\mathcal{R}{:}A\vee B,\mathcal{R}_{A*B}{:}B\mid\ldots\mid\mathcal{S}_n$$

Suppose also that σ_A is a closing substitution for the leftmost sequent $\mathcal{S}_A,\mathcal{R}{:}A\vee B,\mathcal{R}_{A*B}{:}A$. Then, the standard branching rule used in a depth first visit of the proof search tree would transform it into the following active frontier:

$$\mathcal{S}_B\sigma_A,FV(\mathcal{R}\sigma_A){:}(A\vee B)\sigma_A,FV(\mathcal{R}_{A*B}\sigma_A){:}B\sigma_A\mid\ldots\mid\mathcal{S}_n\sigma_A$$

Thus, the rigid variables of $B\sigma_A$ in the next sequent to be considered are those of $FV(\mathcal{R}_{A*B}\sigma_A)$ which means $FV(\mathcal{R}\sigma_A)\cup(FV(A\sigma_A)\cap FV(B\sigma_A))$.

In contrast, the optimized branching rule would yield the following frontier (again deleting closed branches):

$$\mathcal{S}_B\sigma_A,FV(\mathcal{R}\sigma_A){:}(A\vee B)\sigma_A,FV(\mathcal{R}\sigma_A){:}B\sigma_A\mid\ldots\mid\mathcal{S}_n\sigma_A$$

Thus, the rigid variables of $B\sigma_A$ in the next sequent considered by a DFS-algorithm for proof search are only $FV(\mathcal{R}\sigma_A)$, usually a subset of the previous case.

For the semantic branching rules of Fig. 2 we can label the right successor of $\mathcal{S}, \mathcal{R}: A \vee B$ in the proof search tree by $\mathcal{S}\sigma_A, FV(\mathcal{R}\sigma_A):B\sigma_A, FV(\mathcal{R}\sigma_A):\neg A\sigma_A$. Again we used $FV(\mathcal{R}\sigma_A)$ and neither $FV(\mathcal{R}_{A*B}\sigma_A)$ nor $FV(\mathcal{R}_A\sigma_A)$.

With semantic branching we do not need to visit the left branch containing $\mathcal{R}_A:A$ first. The search process could proceed through the right branch containing $\mathcal{R}_{A*B}:B$ and $\mathcal{R}_A:\neg A$ first. Indeed this could be the best choice since B has more universal variables. Then we can reformulate the branching rule as follows: let $\mathcal{S}, \mathcal{R}:A \vee B$ be a sequent and let $\sigma_{B,\neg A}$ be a closing substitution for the proof search tree whose root is labelled by $\mathcal{S}, \mathcal{R}:A \vee B, \mathcal{R}_{A*B}:B, \mathcal{R}_A:\neg A$. Then the right successor of $\mathcal{S}, \mathcal{R}:A \vee B$ in the proof search tree can be $\mathcal{S}\sigma_{B,\neg A}, FV(\mathcal{R}\sigma_{B,\neg A}):A\sigma_{B,\neg A}$.

6 An efficient yet lean implementation

The calculus with renaming, first order reduction and simplification can be easily implemented in sicstus prolog. We only sketch key issues of the prover Beatrix, which has been designed as a lean implementation of the calculus along the line of Beckert and Possega [7].

The first tricky bit is the implementation of renaming modulo rigid variables and of the "one-sided-restricted-unification" denoted by $A = B\sigma_{\overline{\mathcal{R}}}$ and used extensively in the first order reduction and simplification operations in Figure 4 and Figure 5.

In a lean prolog implementation we use prolog variables for first order variables and this makes it possible a simple implementation of the operation of renaming of a formula A modulo a set of rigid variables $\mathcal{R}$:

```
rename_fml(A,R,NewA):- copy_term(A:R,NewA:R).
```

Restricted unification $A = B\sigma_{\overline{\mathcal{R}}}$ are implemented using a predicate of the standard `terms` library of sicstus:

```
restricted-unify(A,B,R):- subsumes(B:R,A:R).
```

The first advantage of having universal variables is that we can give precedence to closing substitutions using only universal variables. Indeed, one of the difficulty faced by any depth-first-algorithm is the choice of a closing substitution when considering sequents one by one: for one sequent we may choose a "locally good but globally wrong" closing substitution that may affect negatively the proof search for another sequent. This happens because the closing substitution may bind the rigid variables shared between the two sequents to the "wrong" values. With closing substitutions with domains of only universal variables, "locally good" means "globally good": by construction that such substations do not affect other branches of the proof search tree. It is therefore important to search for such closing substitution first. This means that given a sequent we look for two formulae $\mathcal{R}_A : A$ and $\mathcal{R}_B : \neg B$ in the sequent such that $A\sigma_{\overline{\mathcal{R}_A}} = B\sigma_{\overline{\mathcal{R}_B}}$. This is implemented with the prolog rule below:

```
unifiable(A:RA,[B:RB|_],_,_):-
  copy_term([A:RA,B:RB],[AN:RAN,BN:RBN]),
  unify_with_occurs_check(AN,BN),
  subsumes_chk(RAN,RA),
  subsumes_chk(RBN,RB).
```

All predicates used in the body come with the standard sicstus distribution.

For the implementation of the main procedure, we search for a proof of a not empty sequent by applying simplification first[2], then first order reduction and finally selecting a formula for reducing it according the rules of the calculus. This can be implemented as

```
prove_seq([A:RA|Seq],Beta,Inst,Lits,Depth):-
  simplify_all(Lits,[A:RA|Seq],SSeq),
  reduce_seq(SSeq,[F:R|RSeq]),
  prove_fml(F:R,RSeq,Beta,Inst,Lits,Depth).
```

where `[A:RA|Seq]` is the set of unprocessed formulae in the sequent, `Beta` is the set of disjunction which have been duplicated in the course of the proof search by the branching rule, `Inst` is used to keep track of the different instances of the disjunctions in `Beta` which have been actually generated by various substitutions, `Lits` is the number of literals so far collected in the current sequent, and `Depth` is the maximum term depth allowed for closing substitutions which bind rigid variables. Instances and term depth are necessary for fairness and termination. As usual, completeness is achieved by iterative deepening over the term depth.

The last tricky bits are the branching rules:

```
prove_fml([or,A|B]:R,Seq,Beta,...):-
 shared_rigid_vars(A,[or|B],R,RA,RB),
 rename_fml(A,RA,NA),
 rename_fml([or|B],RB,[or|NB]),
 rename_fml([or,A|B],R,[or|D]),
 prove_seq([NA:RA|Seq],[[or|D]:R|Beta],...),
 normalise([[or|NB]:RB|Seq],NewSeq),
 normalise([[or|D]:R|Beta],NewBeta),
 ...
 prove_seq(NewSeq,NewBeta,...).
```

[2]The simplified formula might be arbitrary whereas we have restricted the simplifying formula to be a literal.

An intuitive explanation of this prolog clause is that the first predicate in the body constructs the set of rigid variables for the left disjunct A and the right disjunct A. They will be different whether we use semantic or symmetric branching, and then the optimized or the unoptimized version. Then we rename immediately the formula A and B and the disjunction $A \vee B$. Then we try to prove the left sequent. Since this operation might bind some rigid variables shared between the two sequents we need to Norma-Lise them before going on with the search on the right.

7 Correctness

For the soundness proof we cannot use the standard techniques, with a rigid assignment to all variables in a proof tree: renaming and the β-rule would be unsound. We must associate the frontier of a proof tree to a formula:

$$\begin{aligned} fml(\mathcal{S}_1 \mid \ldots \mid \mathcal{S}_n) &= fml(\mathcal{S}_1) \vee \ldots \vee fml(\mathcal{S}_n) \\ fml(\mathcal{S}) &= \forall x_1 \ldots x_n . \bigwedge_i A_i \end{aligned}$$

where $x_k \in FV(\mathcal{S}) \setminus \bigcup_i \mathcal{R}_i$ and $\mathcal{R}_i{:}A_i \in \mathcal{S}$.

Then we must prove that, given the frontier of a proof tree $\mathcal{S}_1 \mid \ldots \mid \mathcal{S}_n$ and the new frontier due to some rule application $\mathcal{S}'_1 \mid \ldots \mid \mathcal{S}'_{n'}$, for all interpretations $\mathcal{I}$ and all assignments $\mathcal{A}$ if $\mathcal{I}, \mathcal{A} \models fml(\mathcal{S}_1 \mid \ldots \mid \mathcal{S}_n)$ then for all assignments $\mathcal{A}'$ extending $\mathcal{A}$ it is $\mathcal{I}, \mathcal{A}' \models fml(\mathcal{S}'_1 \mid \ldots \mid \mathcal{S}'_{n'})$.

The case for the *ren*-rule is then easy and the case for β-rule stems from the observation that $\forall x.(A \vee B)$ is equivalent to $(\forall x.A) \vee B$ if $x \notin FV(B)$. The soundness of the *simp*-rule and the reduction rules is similar: in the replacement steps we only unify universal variables.

The optimized version of either branching rules can be proven sound by a cut and paste argument over the unoptimized proof of a formula.

8 Comparison with related works

Already at the propositional level the calculus for simplification subsumes a number of related works. Here we just list some of them and refer to Massacci [23] for further analysis.

For instance the DPLL-unit rule [12, 11], the β_i^c rules for KE by D'Agostino and Mondadori [9], modus ponens, tollens etc. of HARP [25] and the $\vee - simp_{0,1}$ rules in KRIS [2] or the boolean constraint propagation operation in DLP [19] and KSAT [16] are all (restricted) instances of the simplification rule (*simp*) immediately followed by $(bool - red)$. Propositional hyper-tableaux [5] can be simulated by our rule.

The traditional way to lift the propositional framework to first order logic is to use the *ground version* of the calculus. We need a γ rule á la Smullyan [29]:

$$\frac{\mathcal{S}, \mathcal{R}{:}A(t)}{\mathcal{S}, \mathcal{R}{:}\forall x.A(x)} \quad \text{where } t \text{ is a ground term}$$

With this rule, no fresh variable is introduced in the proof search and hence we have no rigid variable. Simplification is simply propositional simplification and we can answer an interesting question: why first order DPLL is so inefficient in comparison with its propositional version?

At first we observe that the first order DPLL procedure is a calculus with off-line skolemisation, simplification, boolean reduction, semantic branching restricted to literals, and the ground γ-rule. Then, recall that using the γ-rule à la Smullyan rather than those with unification leads to a *non-elementary* slow down [3]. The exponential speed-up due to propositional simplification cannot compensate it.

The tableau calculus of Beckert and Possega [7] with universal variables is an instance of the calculus with renaming but without simplification where

1. in the β rule $\mathcal{R}_{A+B} \doteq \mathcal{R} \cup FV(A) \cup FV(B)$,
2. renaming is applied only to the literals in $\mathcal{S}$ before any β-rule.

This general technique also subsumes the use of conjunctive super-formulae by Degtyarev and Voronkov [13] which is based on restriction (1).

To ease the comparison, it is useful to consider the following fragment of a proof tree in which we apply a β rule and then immediately apply renaming to all formulae of both branches.

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{S}'';\mathcal{R}_{A*B}{:}B\eta_{\overline{\mathcal{R}_{A*B}}}}{\mathcal{S}'', \mathcal{R}_{A*B}{:}A}\, ren}{\mathcal{S}', \mathcal{R}_{A*B}{:}A}\, ren \qquad \mathcal{S}', \beta_2{:}\mathcal{R}_D}{\mathcal{S}', \mathcal{R}{:}A \vee B}\, \beta}{\mathcal{S}, \mathcal{R}{:}A \vee B}\, ren$$

where $\mathcal{S}' = \{\varphi_i \eta_{i\overline{\mathcal{R}}} \mid \varphi_i \in \mathcal{S}\}$ is obtained by a sequence of renaming steps and $\mathcal{S}''$ is obtained from $\mathcal{S}'$ in a similar way, using $\mathcal{R}_{A*B}$ rather than $\mathcal{R}$.

This proof fragment can be "approximated" with the following one:

$$\frac{\mathcal{S}\eta'_{\overline{\mathcal{R}}}, \mathcal{R}_{A*B}{:}A\eta_{\overline{\mathcal{R}_{A*B}}} \quad \mathcal{S}, \mathcal{R}_{A*B}{:}B}{\mathcal{S}, A \vee B{:}\mathcal{R}}\, \beta + ren$$

In practice, we apply one renaming to the whole sequent rather than renaming each formula. Notice that we apply $\eta'_{\overline{\mathcal{R}}}$ and not $\eta'_{\overline{\mathcal{R}_{A*B}}}$.

The net effect is to *separate* the universal variables of $A \vee B$ from those of $\mathcal{S}$ and the universal variables of the two branches. In this way we avoid the potential waste of an universal variable which may be present in formulae like $\forall x.P(x) \wedge (Q(x,y) \vee R(x,y))$. Clearly the x in $P(x)$ is universal and so is the the x in $Q(x,y) \vee R(x,y)$. However, without renaming, the application of the β-rule makes x a rigid variable. This is what happen if we use Beckert and Possega handling of universal variables which limits renaming to literals.

A comparison with the first order hyper-tableau expansion step by Baumgarten et al. [5, 4] is interesting: the idea of a purifying (grounding) substitution used in the extension step of hyper-tableaux can be casted in our framework with a particular search strategy and a general constraint on the final frontier of our proof tree: all rigid variables must be grounded by the closing substitution $\sigma_{\mathcal{T}}$.

For sake of example, assume that we have a binary clause $A \vee B$ and that we can apply to $A \vee B$ a substitution σ such that $FV(A\sigma)$ and $FV(B\sigma)$ are disjoint. Then the hyper-tableaux extension step would yield two branches one with $A\sigma$ and one with $B\sigma$, the instantiated formula $(A \vee B)\sigma$ would be discarded and we would only need to remember the initial clause $A \vee B$.

This is also the case for our branching rule. Indeed, we can strengthen proposition 1 as follows:

Proposition 2 *In the symmetric branching rules of Figure 2, the disjunction $\mathcal{R} : A \vee B$ can be deleted from both consequents if $FV(A) \cap FV(B) = \emptyset$.*

Non-clausal resolution by Manna and Waldinger [22] or the recent version by Björner et al. [8] shares some features of our proposal. Indeed, at propositional level unit non-clausal resolution is a particular instance of simplification. However, when performing a first-order non-clausal resolution step between two subformulae, the unifying substitution must be applied to the main formulae (tableau in the terminology of Manna and Waldinger). In contrast, our simplification rules allow for local substitutions and even different substitutions within the same formula.

The other close cousin of our approach is the FDPLL procedure developed by Baumgarten for the latest CASC competition. However, there are no simplification rules in his calculus which is limited to the CNF format and the usage of universal variables is more restricted.

9 Conclusion

In this paper we have presented a general framework for the introduction of full first order reduction rules for tableau and DPLL-like calculi which can make them fully competitive with resolution in the first order theorem proving setting.

The reduction rules which have been proposed play the same role of the unit rule for propositional DPLL, of subsumption inferences for resolution, and of a number of boolean and modal constraint propagation techniques applied in the description and modal logic communities. Our uniform framework thus subsumes a number of techniques for the improvement of tableau-based methods. Moreover, a prototype implementation is easily implemented using sicstus prolog, along the line of leanTAP [7].

Still, a doubt remains: the "beauty" of the tableau calculi and DPLL is their simplicity, so by adding these reduction rules for constraint propagation are not we abandoning the very calculus we try to enhance?

The author believes that the best answer is probably a question: is resolution with subsumption no longer resolution?

Acknowledgments

I would like to acknowledge many fruitful discussions with B. Beckert, L. Carlucci Aiello, F. Donini, U. Furbach, E. Giunchiglia, F. Giunchiglia, L. Paulson, and F. Pirri. I would like to thank P. Baumgarten for commenting on an earlier draft of this paper and for explaining to me the tricky bits of his calculus underlying FDPLL.

This work started while I was at the University of Cambridge (UK) and the Dipartimento of Informatica and Sistemistica of the University of Roma I "La Sapienza". It has been finalized thanks to a CNR short-term mobility grant at the University of Koblenz-Landau (D).

References

[1] Astrachan, O., and Stickel, M. Caching and lemmaizing in model elimination theorem provers. In *Proc. of the 11th International Conference on Automated Deduction (CADE-90)* (1992), vol. 607 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 224–238.

[2] Baader, F., and Hollunder, B. A terminological knowledge representation system with

complete inference algorithms. In *Proc. of the International Workshop on Processing Declarative Knowldege (PDK-91)* (1991), H. Boley and M. Richter, Eds., vol. 567 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 67–86.

[3] BAAZ, M., AND FERMÜLLER, C. G. Non-elementary speedups between different versions of tableaux. In *Proc. of the 4th Workshop on Theorem Proving with Analytic Tableaux and Related Methods* (1995), P. Baumgartner, R. Hähnle, and J. Possega, Eds., vol. 918 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 217–230.

[4] BAUMGARTNER, P. Hyper tableau - the next generation. In *Proc. of the International Conference on Analytic Tableaux and Related Methods (TABLEAUX-98)* (1998), vol. 1397 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 60–76.

[5] BAUMGARTNER, P., FURBACH, U., AND NIEMELÄ, I. Hyper tableaux. In *Proc. of the 6th European Workshop on Logics in Artificial Intelligence (JELIA-96)* (1996), J. Alferes, L. Pereira, and E. Orlowska, Eds., vol. 1126 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 1–17.

[6] BECKERT, B., AND HÄNLE, R. An improved method for adding equality to free variable semantic tableaux. In *Proc. of the 11th International Conference on Automated Deduction (CADE-90)* (1992), D. Kapur, Ed., vol. 607 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 507–521.

[7] BECKERT, B., AND POSSEGA, J. `leanTAP`: Lean tableau-based deduction. *J. of Automated Reasoning 15*, 3 (1995), 339–358. A preliminary version appeared in *Proc. of the 12th International Conference on Automated Deduction (CADE-94)* (1994), A. Bundy, Ed., vol. 814 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag.

[8] BJÖRNER, N., STICKEL, M. E., AND URIBE, T. E. A practical integration of first-order reasoning and decision procedures. In *Proc. of the 14th International Conference on Automated Deduction (CADE-97)* (1997), Lecture Notes in Artificial Intelligence, Springer-Verlag, pp. 101–115.

[9] D'AGOSTINO, M., AND MONDADORI, M. The taming of the cut. *J. of Logic and Computation 4*, 3 (1994), 285–319.

[10] DAVIS, M. Eliminating the irrelevant from mechanical proofs. In *Proc. of the 15th Symposium in Applied Mathematics* (1963), N. Metropolis, A. H. Taub, J. Todd, and C. Tompkins, Eds., American Mathematical Society, pp. 15–30.

[11] DAVIS, M., LONGEMANN, G., AND LOVELAND, D. A machine program for theorem-proving. *Communications of the ACM 5*, 7 (1962), 394–397.

[12] DAVIS, M., AND PUTNAM, H. A computing procedure for quantificational theory. *J. of the ACM 7*, 3 (1960), 201–215.

[13] DEGTYAREV, A., AND VORONKOV, A. Equality elimination for the tableau method. In *Proc. of the International Symposium on the Design and Implementation of Symbolic Computation Systems (DISCO-96)* (1996), J. Calmet and C. Limongelli, Eds., vol. 1128 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 46–60.

[14] EISINGER, N., OHLBACH, H. J., AND PRÄCKLEIN, A. Reduction rules for resolution-based systems. *Artificial Intelligence 50*, 2 (1991), 141–181.

[15] FITTING, M. *First Order Logic and Automated Theorem Proving.* Springer-Verlag, 1990.

[16] GIUNCHIGLIA, F., AND SEBASTIANI, R. Building decision procedures for modal logics from propositional decision procedures - the case study of modal k. In *Proc. of the 13th International Conference on Automated Deduction (CADE-96)* (1996), M. A. McRobbie and J. K. Slaney, Eds., vol. 1104 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 583–597.

[17] HÄHNLE, R., AND SCHMITT, P. The liberalized δ-rule in free variable semantic tableaux. *J. of Automated Reasoning 13*, 2 (1994), 211–222.

[18] HORROCKS, I. Using an expressive description logic: FaCT or finction? In *Proc. of the 7th International Conference on the Principles of Knowledge Representation and Reasoning (KR-96)* (1998), Morgan Kaufmann, Los Altos, pp. 636–647.

[19] HORROCKS, I., AND PATEL-SCHNEIDER, P. F. Optimising propositional modal satisfiability for description logic subsumption. In *Proc. of the International Conference on Artificial Intelligence and Symbolic Computation (AISC-98)* (1998), vol. 1476 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 234–246.

[20] HUSTADT, U., AND SCHMIDT, R. A. On evaluating decision procedure for modal logic. In *Proc. of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)* (1997), M. Pollack, Ed., pp. 202–207.

[21] LETZ, R., MAYR, K., AND GOLLER, C. Controlled integration of the cut rule into connection tableau calculi. *J. of Automated Reasoning 13*, 3 (1994), 297–337.

[22] MANNA, Z., AND WALDINGER, R. Fundamentals of deductive program synthesis. *IEEE Transactions on Software Engineering 18*, 8 (1992), 674–704.

[23] MASSACCI, F. Simplification: A general constraint propagation technique for propositional and modal tableaux. In *Proc. of the International Conference on Analytic Tableaux and Related Methods (TABLEAUX-98)* (1998), H. de Swart, Ed., vol. 1397 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 217–231. An extended version is available as Technical Report 424, Computer Laboratory, University of Cambridge (UK), 1997.

[24] MASSACCI, F. Design and results of the tableaux-99 non-classical (modal) systems comparison. In *Proc. of the International Conference on Analytic Tableaux and Related Methods (TABLEAUX-99)* (1999), N. Murray, Ed., vol. 1617 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 14–18.

[25] OPPACHER, F., AND SUEN, E. HARP: a tableau-based theorem prover. *J. of Automated Reasoning 4* (1988), 69–100.

[26] PRAWITZ, D. An improved proof procedure. *Theoria 26* (1960), 102–139.

[27] ROBINSON, J. A. A machine oriented logic based on the resolution principle. *J. of the ACM 12*, 1 (1965), 23–41.

[28] SHANIN, N., DAVYDOV, G., YU., M. S., MINTS, G. E., ORENKOV, V. P., AND SLISENKO, A. O. An algorithm for machine search of a natural logical deduction in a propositional calculus. In *Automation of Reasoning (Classical Papers on Computational Logic. Vol. 1 (1957-1966)*, J. Siekmann and G. Wrightson, Eds. Springer-Verlag, 1983.

[29] SMULLYAN, R. M. *First Order Logic.* Springer-Verlag, 1968. Republished by Dover, New York, in 1995.

[30] VORONKOV, A. The anatomy of vampire (implementing bottom-up procedures with code trees). *J. of Automated Reasoning 15*, 2 (1995), 237–265.

[31] VORONKOV, A. Strategies in rigid-variable methods. In *Proc. of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)* (1997), pp. 114–121.

[32] WALLACE, K., AND WRIGHTSON, G. Regressive merging in model elimination tableau-based theorem provers. *J. of the Interest Group in Pure and Applied Logic 3*, 6 (1995), 921–937.

[33] ZHANG, H., AND STICKEL, M. E. An efficient algorithm for unit-propagation. In *Proc. of the 4th International Symposium on Artificial Intelligence and Mathematics* (1996).

Deciding K using ⅄

Andrei Voronkov
Department of Computer Science
The University of Manchester
voronkov@cs.man.ac.uk

Abstract

We present a bottom-up decision procedure for the propositional modal logic K based on the inverse method. The procedure is based on the "inverted" version ⅄ of a sequent calculus for K. The first (essentially nonoptimized) implementation of the procedure shows its competitiveness with the best state-of-the-art systems.

Note. The use of symbols like ⅄ is not directly supported by the standard text-processing systems. When citing this paper, we suggest to use the title *"Deciding* K *using* K*-inverse"* or *simply "Deciding* K *using* K*"*. In LaTeX the symbol ⅄ can be reproduced by using the commands

```
\usepackage{graphicx}
...
\newcommand{\inverseK}{\rotatebox[]{180}{K}}
```

1 Introduction

Nonclassical (propositional) logics play an increasing role in computer science. They are used in model checking, verification, and knowledge representation. Traditional decision procedures for these logics use semantic tableaux [Baader and Hollunder 1991, Balsiger, Heuerding and Schwendimann 1998], optimized tableaux [Horrocks 1998, Patel-Schneider 1998], SAT-based methods [Giunchiglia and Sebastiani 1996, Giunchiglia, Giunchiglia, Sebastiani and Tacchella 1998, Tacchella 1999], or highly optimized translations into first-order classical logic [Hustadt and Schmidt 1997, Schmidt 1998].

In this paper we present a bottom-up decision procedure for propositional modal logic K based on the inverse method. Our techniques can be easily modified to its multi-modal variant K(m) with m modalities, but we restrict ourselves to K for simplicity. The modal logic K(m) is important for several reasons, mainly because this logic is a syntactic variant of the knowledge representation formalism $\mathcal{ALC}$. Our experiments demonstrate that bottom-up procedures can be competitive with the best state-of-the-art methods for K.

We believe that the technique described in our paper opens an interesting research path in the implementation of decision procedures for other important nonclassical logics, such as temporal and description logics.

This paper is organized as follows. In Section 2 we introduce the modal logic K and discuss traditional proof-search procedures for this logic. In Section 3 we define the logical calculus ⅄ used by the inverse method proof procedures, formulate several properties of this calculus and show how proof-search can be organized based on these properties. In Section 4 we discuss several redundancies in ⅄. Exploitation of the redundancies results in more efficient proof procedures. In Section 5 we briefly describe the implementation of the inverse method in our system K⅄ and provide experimental evidences showing competitiveness of K⅄ with the state-of-the-art systems. In Section 6 we consider related work, mainly related to theorem proving by the inverse method. Finally, in Section 7 we discuss future research directions.

Due to the space restrictions we omit proofs in this paper. The full proofs are included in [Voronkov 2000] available on the Web.

2 Preliminaries

In this paper we consider propositional modal logic K. K is the minimal modal logic, i.e. the set of

modal formulas valid in all Kripke models. For the discussion of modal logics and Kripke models see any standard references [e.g. Fitting 1983, Wallen 1990]. In this paper, we will speak about (un)satisfiability of formulas instead of the dual notion of *validity*. It is known that the satisfiability problem for K is PSPACE-complete. A number of interesting properties of the knowledge representation formalism $\mathcal{ALC}$, like concept subsumption, can be represented as statements about (un)satisfiability of formulas in the multimodal version K(m) of K.

There are several characterizations of satisfiability for K.

1. The characterization of satisfiability based on Kripke structures gives rise to two kinds of methods: SAT-based methods, in which satisfiability-checking procedures for propositional classical logic are extended to Kripke structures, and translation-based methods, in which satisfiability of a propositional modal formula is encoded as a first-order formula of a special form, and specialized decision procedures for first-order formulas are used to check the satisfiability of this first-order formula.
2. An alternative characterization of satisfiability by using calculi of a special form, called tableau calculi, gives rise to tableau-based procedures, in which the satisfiability of a formula is checked in a top-down way, by inspecting the inference rules that can be used to derive this formula.

There exist other characterizations of satisfiability for modal logics, for example the matrix characterizations developed in [Wallen 1990], but they have not so far lead to efficient implementations.

Our method, called the *inverse method* following Maslov [1983], uses the characterization of satisfiability in terms of the so-called sequent calculi, developed for classical and nonclassical logics by Gentzen [1934], [see also Goble 1974]. Tableau calculi can often be reformulated as sequent calculi.

In this section we introduce two sequent calculi for K and show how they can be used for proof-search by the tableau method, in the next section we will introduce a similar calculus more suitable for the inverse method.

A *literal* is a propositional variable or its negation. We only consider *modal formulas in negation normal form*, i.e. formulas built from literals using $\wedge$, $\vee$, $\Box$ and $\Diamond$. A formula is called *satisfiable* if it is true in at least one Kripke model. Satisfiability in K can be characterized using a *sequent calculus* introduced below.

We denote propositional variables by lower-case letters, and formulas by upper-case letters. We will use the formula

$$E = \Box a \wedge \Box(\neg a \vee b) \wedge \Diamond \neg b \wedge \Diamond b \qquad (1)$$

for nearly all examples of this paper. We consider conjunction and disjunction right-associative. For example, formula (1) denotes $\Box a \wedge (\Box(\neg a \vee b) \wedge (\Diamond \neg b \wedge \Diamond b))$.

A *sequent* is a finite multiset of formulas. We denote sequents by Greek letters Γ and Δ. We denote the multiset union of two multisets Γ and Δ by Γ, Δ and the addition of an element A to a multiset Γ by Γ, A. For a sequent $\Gamma = A_1, \ldots, A_n$, we denote by $\Diamond\Gamma$ the sequent $\Diamond A_1, \ldots, \Diamond A_n$. A sequent $A_1, \ldots, A_n$ is called *satisfiable* if there exists a Kripke model in which all formulas $A_1, \ldots, A_n$ are true.

It is well-known that satisfiability in K can be characterized using a sequent calculus, i.e. a calculus of a special form that derives sequents.

The *sequent calculus* Seq is given in Figure 1. We call a

- *inference* in Seq any instance of an inference rule;
- *derivation of a sequent* Γ any tree formed by inferences and having Γ as the root;
- *refutation of a sequent* Γ any derivation of Γ whose leaves are axioms.

We will use the notions of inference, derivation and refutation for all other calculi introduced in this paper.

An example refutation in this calculus is given in Figure 2. The connection between derivability in Seq and satisfiability of sequents is give by the following Completeness Theorem.

THEOREM 2.1 (Completeness of Seq) *A sequent S is unsatisfiable if and only if it has a refutation in* Seq.

The proof of this theorem may be found in standard textbooks [e.g. Fitting 1983] or in [Goble 1974].

Tableau-based procedures start with the goal formula, and apply the inference rules of Seq in the direction from the conclusion to the premises. For example, the tableau method can obtain the derivation of Figure 2 using the following sequence of steps:

axioms: $\Gamma, p, \neg p$

$$\frac{\Gamma, A, B}{\Gamma, A \wedge B}\ (\wedge) \qquad \frac{\Gamma, A \quad \Gamma, B}{\Gamma, A \vee B}\ (\vee) \qquad \frac{\Delta, A}{\Gamma, \Box\Delta, \Diamond A}\ (\Diamond)$$

Figure 1: Sequent calculus Seq

$$\frac{\dfrac{\dfrac{a, \neg a, \neg b \quad a, b, \neg b}{a, \neg a \vee b, \neg b}\ (\vee)}{\Box a, \Box(\neg a \vee b), \Diamond\neg b, \Diamond b}\ (\Diamond)}{\Box a \wedge \Box(\neg a \vee b) \wedge \Diamond\neg b \wedge \Diamond b}\ (\wedge), (\wedge), (\wedge)$$

A sequence of inferences $(\wedge)$ is shown as one inference.

Figure 2: Derivation of Formula (1) in Seq

$$\Box a \wedge \Box(\neg a \vee b) \wedge \Diamond\neg b \wedge \Diamond b \quad \Rightarrow$$

$$\frac{\Box a, \Box(\neg a \vee b), \Diamond\neg b, \Diamond b}{\Box a \wedge \Box(\neg a \vee b) \wedge \Diamond\neg b \wedge \Diamond b}\ (\wedge), (\wedge), (\wedge) \quad \Rightarrow$$

$$\frac{\dfrac{a, \neg a \vee b, \neg b}{\Box a, \Box(\neg a \vee b), \Diamond\neg b, \Diamond b}\ (\Diamond)}{\Box a \wedge \Box(\neg a \vee b) \wedge \Diamond\neg b \wedge \Diamond b}\ (\wedge), (\wedge), (\wedge) \quad \Rightarrow$$

$$\frac{\dfrac{\dfrac{a, \neg a, \neg b \quad a, b, \neg b}{a, \neg a \vee b, \neg b}\ (\vee)}{\Box a, \Box(\neg a \vee b), \Diamond\neg b, \Diamond b}\ (\Diamond)}{\Box a \wedge \Box(\neg a \vee b) \wedge \Diamond\neg b \wedge \Diamond b}\ (\wedge), (\wedge), (\wedge)$$

Tableau-based procedures are easy to design. To make them more efficient, a number of optimizations of two kinds are used. One kind is logical optimizations that exploit proof-theoretic properties of Seq, for example invertibility of inference rules, another kind are optimizations of the backtrack search algorithms used by such procedures.

The inverse method discussed in this paper checks goal formulas for satisfiability using proof-search in sequent calculi in the inverse direction, i.e., from the premises of the rules to their conclusions (hence the name "inverse"). First, it builds axioms of the sequent calculus, and then derives more and more complex sequents until a refutation is obtained. For the example above, the inverse method would find a derivation using the following sequence of steps:

$$a, \neg a, \neg b \quad \Rightarrow \quad a, \neg a, \neg b \quad a, b, \neg b \quad \Rightarrow$$

$$\frac{a, \neg a, \neg b \quad a, b, \neg b}{a, \neg a \vee b, \neg b}\ (\vee) \quad \Rightarrow$$

$$\frac{\dfrac{a, \neg a, \neg b \quad a, b, \neg b}{a, \neg a \vee b, \neg b}\ (\vee)}{\Box a, \Box(\neg a \vee b), \Diamond\neg b, \Diamond b}\ (\Diamond) \quad \Rightarrow$$

The idea of the inverse method may seem strange, for example there is an infinite number of possibilities for choosing the axioms, but it can be made feasible by exploiting some proof-theoretic properties of the sequent calculus, most notably the ***subformula property***:

LEMMA 2.2 *Every formula occurring in a derivation of a sequent Γ consists of subformulas of formulas of Γ.*

The subformula property obviously follows from the completeness theorem for Seq.

The first immediate consequence of the subformula property is that amongst the axioms $\Gamma, p, \neg p$ of Seq only some of them can be used to prove a given goal formula G, namely those in which the literals p, $\neg p$ and every formula in Γ are subformulas of G. Then, we can choose the single representative $p, \neg p$ among all sequents of the form $\Gamma, p, \neg p$. This idea requires a reformulation of Seq in a suitable way explained in the next section.

3 The inverse calculus $\mathfrak{K}$

In this section we introduce a sequent calculus suited for a bottom-up proof-search. Such a calculus can be obtained in a straightforward way by modifying Seq.

In the inverse method, the sequent calculus depends on the goal formula G. Such a calculus, denoted by $\mathfrak{K}_G$ and an example derivation in this calculus are shown in Figures 3 and 4. Compare this derivation with that of Figure 2.

The completeness of the inverse method is formulated as follows:

$$\text{axioms: } p, \neg p \qquad \frac{\Gamma, A, A}{\Gamma, A}\ (C)$$

$$\frac{\Gamma, A}{\Gamma, A \wedge B}\ (\wedge_l) \qquad \frac{\Gamma, B}{\Gamma, A \wedge B}\ (\wedge_r)$$

$$\frac{\Gamma, A \quad \Delta, B}{\Gamma, \Delta, A \vee B}\ (\vee)$$

$$\frac{\Gamma, A}{\Box\Gamma, \Diamond A}\ (\Diamond) \qquad \frac{\Gamma}{\Box\Gamma, \Diamond A}\ (\Diamond^+)$$

In the rules, all formulas occurring in the rules are subformulas of G.

Figure 3: Sequent calculus $\mathcal{K}_G$

$$\dfrac{\dfrac{\dfrac{\dfrac{a, \neg a \quad b, \neg b}{a, \neg a \vee b, \neg b}\ (\vee)}{\Box a, \Box(\neg a \vee b), \Diamond \neg b}\ (\Diamond)}{\Box a,\ \Box(\neg a \vee b) \wedge \Diamond \neg b \wedge \Diamond b,\ \Box(\neg a \vee b) \wedge \Diamond \neg b \wedge \Diamond b}\ (\wedge), (\wedge), (\wedge)}{\dfrac{\Box a, \Box(\neg a \vee b) \wedge \Diamond \neg b \wedge \Diamond b}{\Box a \wedge \Box(\neg a \vee b) \wedge \Diamond \neg b \wedge \Diamond b}\ (\wedge), (\wedge), (C)}\ (C)$$

Some sequences of inferences are shown as one inference.

Figure 4: Derivation in $\mathcal{K}_E$

THEOREM 3.1 (Completeness of $\mathcal{K}_G$) *A formula G is unsatisfiable if and only if it has a refutation in $\mathcal{K}_G$.*

The proof of this statement is based the following lemma, called the *subsequent lemma*.

LEMMA 3.2 (Subsequent Lemma) *Let a sequent Γ have a refutation in* Seq *and consists of subformulas of a formula G. Then there exists a subsequent $\Delta \subseteq \Gamma$ that has a refutation in $\mathcal{K}_G$.*

The subsequent lemma can be proved by a straightforward induction on the size of a formula. In this paper we will avoid writing proofs. Instead, we will try to explain ideas of our method in an elementary way. Proofs can be found in [Voronkov 2000].

When we discuss redundancies in $\mathcal{K}_G$ below, we will assert several properties of sequents and derivations in Seq, but not in $\mathcal{K}_G$. Th Subsequent Lemma allows us to transform such properties of Seq into redundancies of $\mathcal{K}_G$.

Note that the calculus $\mathcal{K}_G$ has the following properties helpful to organize bottom-up proof-search procedures.

1. Every formula occurring in a derivation is a subformula of G.
2. There exists a finite number of axioms.
3. For every sequent Γ, there exists only a finite number of ways of applying $(\wedge_l)$, $(\wedge_r)$, $(\Diamond)$, $(\Diamond^+)$, or (C) to Γ.
4. For every sequents Γ and Δ, there exists only a finite number of ways of applying $(\vee)$ to Γ and Δ.

A typical proof procedure based on the inverse method uses these properties and works roughly as follows.

1. Let S be a set of sequents, initially $\{p, \neg p \mid p, \neg p$ are subformulas of $G\}$.
2. Repeatedly apply all possible inferences to sequents in S, adding their conclusions to S until no new sequents can be obtained or a refutation of G is found.

However, such naive procedures are hopelessly inefficient, unless they are augmented with powerful *redundancy criteria*. There are two kinds of redundancy criteria, allowing us to get rid of *redundant sequents* and *redundant inferences*. Intuitively, a sequent is redundant if it can be removed from the search space without loosing completeness. An inference is redundant if the calculus obtained by dropping this inference is still complete. Very powerful techniques for proving redundancies have been developed in resolution-based calculi for classical logic, [see e.g. Bachmair and Ganzinger 1999, Niewenhuis and Rubio 1999]. The use of such powerful criteria is one of the reasons for the efficiency of the system **TA** [Hustadt and Schmidt 1997, Schmidt 1998] that translates formula into a fragment of first-order logic and uses a first-order theorem prover SPASS that implements many redundancy criteria.

Redundancy criteria can be incompatible. For example, it is possible that some sequents Γ and Δ are redundant, but every refutation contains at least one of these sequents. Therefore, one usually proves completeness for a collection of redundancy criteria. If we prove several redundancy criteria for a calculus, the proof-search by the inverse method works as follows.

1. Let S be a set of sequents, initially

$\{p, \neg p \mid p, \neg p$ are subformulas of $G\}$ *less the redundant sequents of this form.*

2. Repeatedly apply all *nonredundant* inferences to sequents in S, adding to S those conclusions of these inferences *that are nonredundant.*

3. Remove all sequents that *become redundant* due to the addition of these conclusions of inferences.

A number of redundancy criteria for various nonclassical logics are proved in [Voronkov 1992] using a rather general technique. A different (but also very general) technique for proving redundancy criteria is presented in [Voronkov 2000]. Our implementation is based on the latter paper. Some of the redundancies discussed in [Voronkov 2000] will be explained here. Most (but not all) of the redundancies used in our system KK can be proved complete by a rather general technique based on proving properties of refutations in Seq and transforming them into redundancy criteria for the inverse method using the Subsequent Lemma.

4 Redundancies

Several redundancy criteria discussed in this paper are based on a particular position of a subformula in the goal formula G. It will be convenient for us to identify subformulas of G with their positions. Therefore, we regard two different occurrences of the same formula in G as different subformulas. In [Voronkov 2000] this is achieved by introducing a special *path calculus* dealing with positions of subformulas in G.

4.1 Modal depth of a subformula

Let us call the *modal depth* of a subformula A of G the number of modalities $\Diamond$ and $\Box$ in G that have A in their scope. For example, if G has the form $\Diamond(\ldots \wedge \Box A)$, then the modal depth of this occurrence of A is two. It is easy to see that there is only one inference rule in Seq that changes the modal depth of subformulas when we go from the conclusion of an inference to its premises, namely $(\Diamond)$, and all formulas in the premise of this rule have the modal depth one greater than those in the conclusion.

LEMMA 4.1 *Let Γ be a sequent occurring in a derivation of G in* Seq. *Then all formulas in Γ have the same modal depth.*

This property immediately gives us the following redundancy criterion: any sequent containing two subformulas of different modal depths is redundant.

4.2 Conflicting subformulas

$\vee$-conflict. Consider the $(\vee)$-rule of Seq:

$$\frac{\Gamma, A \quad \Gamma, B}{\Gamma, A \vee B} \ (\vee).$$

One can prove that the sequents above this inference can either contain A, or B but never both (recall that we identify subformulas with their occurrences in G and treat two different occurrences as different subformulas). Moreover, the same holds for the subformulas of A and B: no sequent in the derivation can contain both a subformula of A and a subformula of B. Therefore, for every subformula $A \vee B$ of Γ we say that each subformula of A *is in* $\vee$*-conflict* with each subformula of B. Then we claim that any sequent that contains a $\vee$-conflicting pair of subformulas is redundant.

$\Diamond$-conflict. A similar observation can be made for some occurrences of subformulas starting with $\Diamond$. Assume that $\Diamond A$ and $\Diamond B$ are two different subformulas of G of the same modal depth. The only rule of Seq that can introduce a modality $\Diamond$ or $\Box$ into a derivation is the rule $(\Diamond)$:

$$\frac{\Delta, C}{\Gamma, \Box\Delta, \Diamond C} \ (\Diamond).$$

But the conclusion of this rule can only contain *one* subformula $\Diamond A$ or $\Diamond B$ but never both. Therefore, we say that $\Diamond A$ and $\Diamond B$, as well as their subformulas, are in $\Diamond$*-conflict* and again treat any sequent containing a conflicting pair of subformulas as redundant.

Of course, these informal considerations do not *prove* that the combination of redundancy criteria considered so far preserves completeness. This is proved formally in every detail in [Voronkov 2000]. Our aim here is to give the reader an informal understanding of the redundancy criteria used in our system. However, the space limitations does not allow us to discuss the proof or rather sophisticated techniques involved in such completeness proofs.

4.3 Subsumption

We say that a sequent Γ *subsumes* a sequent Δ if $\Gamma \subseteq \Delta$[1]. *Subsumption* is the redundancy criterion that allows one to remove from the search space sequents

[1]This notion of subsumption introduced in resolution-based theorem proving by Robinson [1965] should not be mixed with the notion of subsumption for description logics that has appeared later.

subsumed by other sequents. Therefore, the subsumed sequents are regarded as redundant.

This notion of subsumption is implemented in version 1.0 of our system KЯ. In fact, we developed an even stronger notion of subsumption in [Voronkov 1992], which is also compatible with all redundancies discussed in this paper, but we could not yet find a way to efficiently implement the stronger notion.

We will informally explain this stronger notion of subsumption. Let A, B be two subformulas of the goal formula (recall that we identify subformulas with their occurrences). We write $A \geq B$ if this occurrence of B is inside of the occurrence of A (this implies that B is a subformula of A). We say that a sequent Γ $\geq$-subsumes a sequent Δ if and only if for every $A \in \Gamma$ there exists $B \in \Delta$ such that $A \geq B$. It is not hard to argue that, if Γ subsumes Δ, then Γ also $\geq$-subsumes Δ. (Indeed, Γ subsumes Δ if and only if for every $A \in \Gamma$ there exists $B \in \Delta$ such that $A = B$, but $A = B$ implies $A \geq B$).

To illustrate the strength of this notion of subsumption let us make an observation. Consider any inference by $(\wedge_l)$:

$$\frac{\Gamma, A}{\Gamma, A \wedge B}\ (\wedge_l).$$

The conclusion of this inference $\geq$-subsumes the premise, since $A \wedge B \geq A$. Therefore, as soon as this inference is performed, the sequent Γ, A can be removed from the search space. A similar observation can be made for any $(\wedge_r)$-inference. Therefore, for any subformula $A \wedge B$ of the goal, its immediate subformulas A and B are "short-living" and any sequent Δ containing any of these subformulas can be replaced by a sequent $\geq$-subsuming Δ. In KЯ this observation is implemented in the following way: the calculus Я_G is modified so it does not contain $(\wedge_l)$ or $(\wedge_r)$-rules at all. Such an optimization in sequent calculi was originally introduced for classical logic in [Voronkov 1985], see also more accessible [. Voronkov 1990]

To prove completeness of Я_G with subsumption a new technique based on so-called *traces* of sequents is developed and explained in [Voronkov 2000].

4.4 Orderings

All redundancy criteria discussed so far introduced redundancy criteria for sequents. Our system KЯ also uses a redundancy criterion for inferences based on orderings of subformulas. Consider any nonmodal subformulas of G, for example $A \wedge B$ and $C \vee D$. Suppose that we have obtained a sequent Γ, A, C containing both A and C. There are at least two kinds of inferences of Я_G applicable to Γ, A, C:

$$\frac{\Gamma, A, C}{\Gamma, A \wedge B, C}\ (\wedge) \quad \text{and} \quad \frac{\Gamma, A, C \quad \Delta, D}{\Gamma, \Delta, A, C \vee D}\ (\vee).$$

Our ordering technique allows one to introduce an ordering $\succ$ on subformulas of G and to only allow the rules be applied to minimal subformulas in a sequent. Therefore, if we have $A \succ C$, then the first of the two inference becomes redundant, since the inference is applied to a nonminimal subformula A. Likewise, if we have $C \succ A$, then the second inference becomes redundant.

The completeness of all redundancy criteria considered before this subsection can be proved in a rather straightforward way using the Subsequent Lemma. However, the completeness of the ordering strategy requires special care and rather involved techniques. For example, not every ordering preserves completeness, but there are orderings strong enough so that exactly one subformula will be selected for inferences in any sequent.

5 Experiments

The calculus Я_G for K with the redundancy criteria mentioned above is implemented in version 1.0 of our system KЯ (read "K-inverse K"). A preliminary much slower version of KЯ was implemented in ML and described in [Voronkov 1999]. The current version 1.0 of KЯ used for benchmarking is implemented in C++ and comprises about 13,000 lines of C++ code, of which 5,000 lines implement a parser generated automatically using the ANTLR parser generator and PCCTS tool.

In this section we consider the results of running KЯ on the collection of tests used for the Tableau'98 competition. We give a comparison with other systems for K. The results for other systems are taken from [Giunchiglia, Giunchiglia and Tacchella 1999, Horrocks, Patel-Schneider and Sebastiani 2000]. The results for other systems were obtained on a computer with the Pentium 350MHz processor using the time limit of 100 seconds. We ran our experiments on a Pentium 450MHz processor and set the limit to 77 seconds, in order to compensate the difference in performance. Some of the results for other systems are from the Tableaux'98 comparison, but some are more recent [see Horrocks et al. 2000].

The experiments were performed on 9 collections of tests. Each collection consists of 21 provable and 21

unprovable formulas of presumably growing difficulty. The results are given in Table 1. p in every column means provable formulas, while n means unprovable. The table contains the number of problems solved by systems within the time limit of 100 seconds. If all 21 problems can be solved, we put > in the table. Table 2 shows the time spent by some systems for solving the most difficult problems in each category.

One can see that KЯ 1.0 is weaker than the two best systems ***SAT** and **DLP**. However, we would like to note that our current implementation is completely nonoptimized as to proof-search strategies and nonlogical optimizations. For example, both ***SAT** and **DLP** have strategies for intelligent backtracking and **DLP** implements tabulation. In addition, all top systems have powerful preprocessors that simplify the input formula, while the current version of KЯ has no preprocessor. KЯ is stronger on this collection of benchmarks than the less optimized implementations **FaCT** of the tableau method, **KSAT** of the SAT-based method, and the earlier versions of **TA** of the first-order translation-based methods. This shows the high potential of future, more optimized versions of KЯ.

Our system is based on a systematic proof procedure without any ad hoc optimizations. We expect a considerable speedup by implementing such optimizations. Our expectations are confirmed by the experience of other provers, whose first versions were very considerably slower than the current versions. Among such future optimizations we consider the following.

1. Implementing a formula preprocessor that makes general simplifications of the goal formula and also some specific simplifications that make the inverse proof search easier.

2. Implementing a greedy search for redundancies, similar to the SPASS theorem prover for first-order classical logic [Weidenbach, Afshordel, Brahm, Cohrs, Engel, Keen, Theobalt and Topic 1999] used as a subsystem in **TA**. This means that we give priority to those inferences that can make some sequents from the search space redundant. Currently, KЯ behaves like a very naive Davis-Putnam procedure for propositional classical logic.

3. Optimizing the system for the cases when propositional reasoning dominates the modal reasoning (cf. the weak performance of KЯ on the *ph* benchmarks in which essentially no modal reasoning is involved).

6 Related works

The inverse method for nonclassical logics. Gentzen [1934] was the first to formulate the inverse method: he has shown how to prove the decidability of intuitionistic propositional logic using the bottom-up search in a sequent calculus based on the subformula property. The remark of Gentzen [1934] remained unnoticed for quite some time. The general interest to the inverse method appeared due to a series of Maslov's works, e.g. [Maslov 1971].

Voronkov [1992] sketches a general scheme of the inverse method for nonclassical logics. Some redundancy criteria considered in this paper are similar to strategies introduced in [Voronkov 1992].

Mints [1993] and Tammet [1994] consider the inverse method for linear logic. Mints [1994] and Tammet [1996] discuss the inverse method for intuitionistic logic, however Tammet's [1996] presentation contains several inaccuracies that are difficult to validate in the absence of proofs. For example, his general scheme of constructing inverse method calculi may result in incomplete calculi, and the intuitionistic calculus of Tammet [1996] is indeed incomplete. The ordering strategy of our paper is similar to a strategy based on the permutability of rules in [Tammet 1996].

Implementation of the inverse method. Tammet [1996] reported some results on the implementation of the inverse method for intuitionistic predicate logic, however it is difficult to comment on his results since calculi underlining his implementation [Tammet 1996] are incomplete.

Our implementation seems to be the first to show that calculi based on forward reasoning may compete with calculi based on backward reasoning for *propositional* logics.

Implementations of other methods. Implementations of other state-of-the-art systems are described in the following papers. SAT-based decision procedures are considered in [Giunchiglia et al. 1998, Tacchella 1999, Giunchiglia et al. 1999]. Hustadt and Schmidt [1997] describe an implementation based on a translation to classical predicate logic and the use of the classical logic theorem prover SPASS [Weidenbach et al. 1999]. Two best tableau-based systems are discussed in [Horrocks and Patel-Schneider 1998, Horrocks 1998, Patel-Schneider 1998]. A comparison of several approaches and a general discussion of evaluation of decision procedures for nonclassical logics can be found in [Hustadt and Schmidt 1997, Hor-

	branch		*d4*		*dum*		*grz*		*lin*		*path*		*ph*		*poly*		*t4p*	
K	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*
leanK 2.0	1	0	1	1	0	0	0	>	>	4	2	0	3	1	2	0	0	0
Crack 1.0	2	1	2	3	3	>	1	>	5	2	2	6	2	3	>	>	1	1
□KE	13	3	13	3	4	4	3	1	>	2	17	5	4	3	17	0	0	3
LWB 1.0	6	7	8	6	13	19	7	13	11	8	12	10	4	8	8	11	8	7
Kris	3	3	8	6	15	>	13	>	6	9	3	11	4	5	11	>	7	5
KSAT	8	8	8	5	11	>	17	>	>	3	4	8	5	5	13	12	10	18
FaCT 1.2	6	4	>	8	>	>	>	>	>	>	7	6	6	7	>	>	>	>
KK	6	7	>	16	>	>	>	>	>	>	15	14	2	2	>	>	>	>
TA	9	9	>	18	>	>	>	>	>	>	20	20	6	9	16	17	>	19
DLP 3.1	19	13	>	>	>	>	>	>	>	>	>	>	7	9	>	>	>	>
***SAT** 1.2	>	12	>	>	>	>	>	>	>	>	>	>	8	12	>	>	>	>

Table 1: Results for K

	***SAT** 1.2		**DLP** 3.2		**TA** 1.4		**KK**	
Test	Size	Time	Size	Time	Size	Time	Size	Time
branch_p	>	0.21	19	46.06	6	65.76	6	29.08
branch_n	12	94.49	13	53.63	6	96.44	7	65.33
d4_p	>	0.06	>	0.05	15	71.11	>	0.09
d4_n	>	2.87	>	1.12	14	44.06	16	69. 17
dum_p	>	0.04	>	0.02	17	64.99	>	0.01
dum_n	>	0.12	>	0.02	16	65.82	>	0
grz_p	>	0.04	>	0.04	>	0.51	>	0.11
grz_n	>	0.01	>	0.05	>	0.33	>	0.08
lin_p	>	0.01	>	0.03	>	9.24	>	0.03
lin_n	>	47.80	>	0.13	>	80.01	>	0.15
path_p	>	0.72	>	0.32	5	25.03	15	36.43
path_n	>	0.96	>	0.36	4	60.84	14	35.92
ph_p	8	48.54	7	10.23	6	43.16	2	0.08
ph_n	12	0.60	>	2.69	9	55.13	2	0.02
poly_p	>	1.73	>	0.11	5	53.48	>	0.54
poly_n	>	2.25	>	0.18	4	9.09	>	0.49
t4p_p	>	0.29	>	0.06	16	88.66	>	0.01
t4p_n	>	1.28	>	0.13	9	87.72	>	0.04

Table 2: Timing Results from [Giunchiglia et al. 1999] and [Horrocks et al. 2000] for ***SAT** (options `-k1 -e -m6`), **DLP**, **TA** and KK.

rocks et al. 2000].

7 Future research

The framework described in this paper is applicable to a number of nonclassical logics, at least K, T, D, $K4$, $T4$, $S4$, their multimodal variants and their variants with global assumptions.

It would be interesting to *extend it to more expressive description logics* [Calvanese, De Giacomo, Lenzerini and Nardi 2000]. An obstacle to such an extension is the way these logics are usually introduced. The technique for designing calculi for various logics developed in the description logic community is essentially oriented to tableau-based procedures. These calculi are formulated in a top-down way and often have explicit annotation for worlds. This simplifies completeness proofs but the resulting calculi do not have obvious proof-theoretical properties to be exploited. As a result, only two kinds of satisfiability checking procedures can be developed: tableau-based ones and semantics-based ones.

To design an inverse-method based calculus one needs to *obtain reformulations of these tableau-oriented calculi in a more traditional way*, where the inference rules are formulated solely in terms of formulas, or otherwise find new techniques for designing bottom-up calculi. Such a logical reformulation is an interesting line of research by itself and can lead to better versions of tableau-based procedures that also exploit proof-theoretical properties of calculi, while the current tableau-based systems are mostly based on various nonlogical optimizations, such as tabulation or intelligent backtracking.

It seems that bottom-up calculi can offer considerable advantages for handling some inference rules. For examples, some description logics have a restricted form of the cut rule:

$$\frac{\Gamma, A \quad \Gamma, \overline{A}}{\Gamma} \ (cut),$$

where A and $\overline{A}$ (equivalent to the negation of A) can be arbitrary subformulas of the goal formula. The problem of using this rule in the direction from the conclusion to the premises is high nondeterminism in selecting the cut formula A. The application in the inverse direction does not meet this problem because Γ, A and $\Gamma, \overline{A}$ must be already derived sequents. Therefore, from the viewpoint of bottom-up calculi there is no essential difference between this rule and, say, the $(\vee)$-rule:

$$\frac{\Gamma, A \quad \Delta, B}{\Gamma, \Delta A \vee B} \ (\vee),$$

in which Γ, A and Δ, B should be already proved sequents.

Another interesting line of research is *combination of top-down and bottom-up proof search*, or *bidirectional search*. In such a combination the inverse method would produce sequents that can be used by the tableau method as additional axioms. This is similar to tabulation, but with an essential difference. The traditional tabulation is used to store *derivable* sequents and checking whether a newly derived sequent is unsatisfiable by verifying whether it is subsumed by a tabulated sequent. If it is, then the newly derived sequent is unsatisfiable, if not, then we have to continue the search for refutation. The inverse method tabulates *all* sequents up to a certain modal depth k, which means that a newly derived sequent of a modal depth $\geq k$ is satisfiable *if and only if* it is not subsumed by any of the tabulated sequent. Therefore, if the subsumption-check fails one should *not* continue search for the proof of this newly derived clause.

A combination of the tableau and the inverse method can also solve the memory problem for the inverse method. On some benchmarks, our implementation of the inverse method uses more than 400M of memory. If one discovers that the memory consumption is too high, one can terminate the inverse proof search and continue the tableau search with the tabulated sequents collected so far.

However, such bidirectional search meets essential difficulties on the theoretical level. The inverse method without redundancies is not efficient at all. The naive tableau proof-search is inefficient as well. When we run bidirectional proof-search, one has to be sure that the optimizations used in the tableau proof-search are compatible with the redundancies used in the inverse proof search. This is a challenging task, since the completeness proofs of the redundancies for the inverse method requires a nontrivial technique.

We feel that a new technique can be used to give model-theoretical completeness proofs of the inverse path calculus with redundancies built in directly into the proof system, similar to completeness proofs for resolution with redundancies [Bachmair and Ganzinger 1997]. More on this point can be found in [Voronkov 2000].

It is not hard to design a procedure which deletes some derived sequents (but when it is necessary proves them again) and runs in polynomial space. However, we

doubt that such a procedure would be any efficient in practice. It is interesting to understand if one can design a bottom-up procedure running in polynomial space and having the following property: a sequent Γ is deleted only when it is redundant, so after deletion Γ should not be derived again.

Conclusion

We introduced and implemented a new method of propositional modal reasoning based on the a special bottom-up version of a sequent calculus. Our experiments show that bottom-up procedures, even implemented in a straightforward way, give a viable alternative to existing methods of automated reasoning for modal and description logics.

We hope that the approach undertaken in this paper gives an interesting pattern for implementing decision procedures for PSPACE-complete problems in general and for more expressive description logics.

Acknowledgments

This paper benefited from numerous discussions with (in the alphabetical order) Enrico Franconi, Enrico Giunchiglia, Fausto Giunchiglia, Ian Horrocks, Fabio Massacci, Ulrike Sattler and Renate Schmidt.

The completeness proofs of various redundancies for the sequent calculus were verified by Anatoli Degtyarev. His enthusiastic remarks about the mathematical content of this research helped me to proceed towards the actual implementation.

The system KH was implemented in a relatively short period of time due to the help from Alexander Riazanov on various aspects of C++ and remarks of Christoph Weidenbach on debugging.

The author is partially supported by a grant from the Faculty of Science and Engineering, University of Manchester.

References

Baader, F. and Hollunder, B. [1991], A terminological knowledge representation system with complete inference algorithms, *in* H. Boley and M. Richter, eds, 'PDK'91', Vol. 567 of *Lecture Notes in Artificial Intelligence*, pp. 67–86.

Bachmair, L. and Ganzinger, H. [1997], A theory of resolution, Research report MPI-I-97-2-005, Max-Planck Institut für Informatik, Saarbrücken, Germany. To appear in Handbook of Automated Reasoning, edited by A. Robinson and A. Voronkov.

Bachmair, L. and Ganzinger, H. [1999], A theory of resolution, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Elsevier Science and MIT Press. To appear.

Balsiger, P., Heuerding, A. and Schwendimann, S. [1998], Logics Workbench 1.0, *in* H. de Swart, ed., 'Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX'98', Vol. 1397 of *Lecture Notes in Computer Science*, Springer Verlag, Oisterwijk, The Netherlands, p. 35.

Calvanese, D., De Giacomo, G., Lenzerini, M. and Nardi, D. [2000], Reasoning in expressive description logics, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Elsevier Science and MIT Press. To appear.

Fitting, M. [1983], *Proof methods for modal and intuitionistic logics*, Vol. 169 of *Synthese Library*, Reidel Publ. Comp.

Gentzen, G. [1934], 'Untersuchungen über das logische Schließen', *Mathematical Zeitschrift* **39**, 176–210, 405–431. Translated as [Gentzen 1969].

Gentzen, G. [1969], Investigations into logical deduction, *in* M. Szabo, ed., 'The Collected Papers of Gerhard Gentzen', North Holland, Amsterdam, pp. 68–131. Originally appeared as [Gentzen 1934].

Giunchiglia, E., Giunchiglia, F., Sebastiani, R. and Tacchella, A. [1998], More evaluation of decision procedures for modal logics, *in* A. Cohn, L. Schubert and S. Shapiro, eds, 'Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR'98)', Morgan Kaufmann, San Francisco, CA, pp. 626–635.

Giunchiglia, E., Giunchiglia, F. and Tacchella, A. [1999], 'SAT-based decision procedures for classical modal logics', *Journal of Automated Reasoning* . To appear in the Special Issues: Satisfiability at the start of the year 2000 (SAT 2000).

Giunchiglia, F. and Sebastiani, R. [1996], Building decision procedures for modal logics from propositional decision procedures: Case study of modal K, *in* M. McRobbie and J. Slaney, eds, 'CADE-13', Vol. 1104 of *Lecture Notes in Computer Science*, pp. 583–597.

Goble, L. [1974], 'Gentzen systems for modal logic', *Notre Dame J. of Formal Logic* **15**, 455–461.

Horrocks, I. [1998], Using an expressive description logic: FaCT or fiction?, *in* A. Cohn, L. Schubert and S. Shapiro, eds, 'Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR'98)', Morgan Kaufmann, San Francisco, CA, pp. 636–647.

Horrocks, I. and Patel-Schneider, P. [1998], FaCT and DLP, *in* H. de Swart, ed., 'Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX'98', Vol. 1397 of *Lecture Notes in Computer Science*, Springer Verlag, Oisterwijk, The Netherlands, pp. 27–30.

Horrocks, I., Patel-Schneider, P. and Sebastiani, R. [2000], 'An analysis of empirical testing for modal decision procedures', *Logic Journal of the IGPL* . submitted.

Hustadt, U. and Schmidt, R. [1997], On evaluating decision procedures for modal logic, *in* 'IJCAI-97', Vol. 1, pp. 202–207.

Maslov, S. [1971], Proof-search strategies for methods of the resolution, *in* B. Meltzer and D. Michie, eds, 'Machine Intelligence', Vol. 6, American Elsevier, pp. 77–90.

Maslov, S. [1983], An inverse method for establishing deducibility of nonprenex formulas of the predicate calculus, *in* J.Siekmann and G.Wrightson, eds, 'Automation of Reasoning (Classical papers on Computational Logic)', Vol. 2, Springer Verlag, pp. 48–54.

Mints, G. [1993], 'Resolution calculus for the first order linear logic', *Journal of Logic, Language and Information* **2**, 58–93.

Mints, G. [1994], Resolution strategies for the intuitionistic logic, *in* 'Constraint Programming', NATO ASI Series F, Springer Verlag, pp. 289–311.

Niewenhuis, R. and Rubio, A. [1999], Paramodulation-based theorem proving, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Elsevier Science and MIT Press. To appear.

Patel-Schneider, P. [1998], DLP system description, *in* E. Franconi, G. Giacomo, R. MacGregor, W. Nutt, C. Welty and F. Sebastiani, eds, 'Collected Papers from the International Description Logic Workshop (DL'98)', Vol. 11 of *CEUR-WS*, pp. 87–89.
URL: *http://SunCITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS*

Robinson, J. [1965], 'A machine-oriented logic based on the resolution principle', *Journal of the Association for Computing Machinery* **12**(1), 23–41. Reprinted as [Robinson 1983].

Robinson, J. [1983], A machine oriented logic based on the resolution principle, *in* J. Siekmann and G. Wrightson, eds, 'Automation of Reasoning. Classical Papers on Computational Logic', Vol. 1965, Springer, pp. 397–415. Originally appeared as [Robinson 1965].

Schmidt, R. [1998], Resolution is a decision procedure for many propositional modal logics, *in* M. Kracht, M. de Rijke, H. Wansing and M. Zakharyaschev, eds, 'Advances in Modal Logic, I', Vol. 87 of *Lecture Notes*, CSLI Publications, Stanford, pp. 189–208.

Tacchella, A. [1999], *SAT system description, *in* P. Lambrix, A. Borgida, M. Lenzerini, R. Möller and P. Patel-Schneider, eds, 'Proceedings of the 1999 International Description Logic Workshop (DL'99)', Vol. 22 of *CEUR-WS*, pp. 142–144.
URL: *http://SunCITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS*

Tammet, T. [1994], 'Proof strategies in linear logic', *Journal of Automated Reasoning* **12**(3), 273–304.

Tammet, T. [1996], A resolution theorem prover for intuitionistic logic, *in* M. McRobbie and J. Slaney, eds, 'CADE-13', Vol. 1104 of *Lecture Notes in Computer Science*, pp. 2–16.

Voronkov, A. [1985], A proof search method (in Russian), Vol. 107 of *Vychislitelnye Sistemy*, Novosibirsk.

Voronkov, A. [1990], A proof-search method for the first order logic, *in* P. Martin-Löf and G. Mintz, eds, 'COLOG'88', Vol. 417 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 327–340.

Voronkov, A. [1992], Theorem proving in non-standard logics based on the inverse method, *in* D. Kapur, ed., '11th International Conference on Automated Deduction', Vol. 607 of *Lecture Notes in Artificial Intelligence*, pp. 648–662.

Voronkov, A. [1999], KK: a theorem prover for K, *in* H. Ganzinger, ed., 'Automated Deduction—CADE-16. 16th International Conference on Automated Deduction', Lecture Notes in Artificial Intelligence, Trento, Italy, pp. 383–387.

Voronkov, A. [2000], How to optimize proof-search in modal logics: a new way of proving redundancy criteria for sequent calculi, Technical Report UMCS-2000-1-1, Department of Computer Science, University of Manchester.
URL: *http://www.cs.man.ac.uk/cstechrep/*

Wallen, L. [1990], *Automated Deduction in Nonclassical Logics*, The MIT Press.

Weidenbach, C., Afshordel, B., Brahm, U., Cohrs, C., Engel, T., Keen, E., Theobalt, C. and Topic, D. [1999], System description: SPASS version 1.0.0, *in* H. Ganzinger, ed., 'Automated Deduction—CADE-16. 16th International Conference on Automated Deduction', Lecture Notes in Artificial Intelligence, Trento, Italy, pp. 378–382.

Uncertainty

A Compositional Structured Query Approach to Automated Inference

Yousri El Fattah
Rockwell Science Center
1049 Camino Dos Rios
Thousand Oaks, CA 91360
yousri@rsc.rockwell.com

Mark Alan Peot
Rockwell Science Center
444 High Street, Suite 400
Palo Alto, CA 94301
peot@rsc.rockwell.com

Abstract

In this paper we propose a compositional structured query approach to automated inference based on a unifying database representation. The representation introduces the notion of weighted relations where special weight attributes are included in the schema to represent probabilities, costs and utilities. The approach also defines generic primitive queries to perform basic inference operations called conditioning, combination, elimination, and solution. The queries are expressed declaratively in the structured query language (SQL) and can be considered as "views" or derived relations defined in terms of other queries or network base relations. We describe an algorithm that compiles from the network database schema a query composition graph (QCG), which is a directed acyclic graph (DAG) whose nodes correspond to queries and the edges denote execution precedence. We give examples to illustrate how the QCG can be reused and incrementally updated for performing reasoning tasks in probabilistic and deterministic networks as well as for accomplishing optimization tasks.

1 Introduction

Relational databases are the most common framework for representing structured data in the real-world. The formal basis for relational database systems is provided by the relational data model (Maier, 1983). In the relational model, a database is a collection of relational tables (or simply relations). The database language SQL is now widely accepted as the standard query language for relational database systems. SQL commands can be used to interact with a database or can be embedded within another programming language to interface to a database. Additional programming extensions have transformed SQL into a complete database programming language.

Relational tables can represent: (a) Discrete probability distributions: a map from input values to probability distributions over a discrete random variable. Example: the result of a weather forecast given the actual weather. (b) Constraints: the set of allowable combinations of input and output values. Example: the output is 1 if the inputs have an odd numbers of 1's. Earlier work (Wen, 1991) examined the relationship between Bayesian networks and databases and (Wong *et al.*, 1995) represented probability requests in SQL queries. Recently a structured modeling language (SML) (El Fattah, 1999) is proposed to represent causal networks in relational databases. The SML language is similar to frame-based BNs (Koller and Pfeffer, 1998). SML differs in that it unites inference, model construction, and representation under the aegis of relational databases.

In this paper we propose a compositional approach to automated inference based on a unifying database representation, similar to that of (Wong *et al.*, 1995). The approach is embedded in our SML language and uses the concept of weighted relations where special weight attributes are included in the schema[1] of the relations to represent probabilities, costs and utilities. We introduce four basic querying operations called conditioning, elimination, combination, and solution queries. We provide a general variable elimination algorithm that compiles the database schema representation of deterministic and probabilistic networks to a

[1] A description of a database to the database management system (DBMS), generated using the data definition language provided by the DBMS. A schema defines attributes of the database, such as tables, columns, properties.

query composition graph (QCG). A QCG is a directed acyclic graph (DAG) whose root nodes are the network base relations and the non-root nodes are instantiations of primitive queries. There is a one-to-one mapping between variables and elimination queries and only the predecessors of the elimination queries include the eliminated variable in their schema. We describe how the QCG can be used in conjunction with answer queries to perform reasoning tasks in probabilistic and deterministic networks, as well as for accomplishing optimization tasks.

By directly expressing our algorithms in the relational language SQL, we are establishing strong connections between automated inference techniques in constraint satisfaction, Bayesian networks, and databases. Embedding automated inference in the realm of databases enables the integration with emerging database technologies, such as data warehousing and data mining. Database management systems are becoming increasingly sophisticated and powerful, and it is most likely that the demands of future applications will continue this trend.

Our proposed QCG provides a meta language for automated inference, where we can adapt computation from one problem to another. Specifically, we can incrementally adapt the QCG to changes in the network representation of a problem or to changes in the evidence. By expressing automated inference methods as SQL querying, we are leveraging all the query processing capabilities and query optimization techniques readily available in relational database management systems. For example, we do not need to write methods for computing consistent solutions or to perform the joining of relational tables. Instead, we can rely on standard SQL query operators.

The structure of the paper is as follows. Section 2 describes the database representation for inference and optimization tasks in probabilistic and deterministic networks. Section 3 introduces generic structured queries in SQL. Section 4 describes the elimination algorithm for compiling the QCG from the network schema. Section 5 describes the algorithm for computing answer queries from the QCG and Section 6 focuses on the problem of finding the MEU for influence diagrams. Section 7 presents discussion and related work and Section 8 gives concluding remarks.

2 Database Representation

This section gives definitions and introduces the notion of weighted relation. We describe the database representation for probabilistic and deterministic networks and state the queries for inference and optimization tasks.

Definition 1 [Relational Databases (Maier, 1983)] Let $U = \{U_1, ..., U_n\}$ be a set of attributes, each with an associated domain. A *relational database scheme* $\mathbf{R}$ over U is a collection of relation schemes $\{R_1, R_2, \ldots, R_p\}$, where $\bigcup_{i=1}^{p} R_i = U$, and $R_i \neq R_j$, if $i \neq j$. A *relational database d on database scheme* $\mathbf{R}$ is a collection of relations $\{r_1, r_2, \ldots, r_p\}$. Each *relation* r_i on relation scheme R_i, written $r_i(R_i)$, is a set of value tuples $\{t_1, t_2, \ldots, t_p\}$ for the attributes in R_i from their respective domains. We write $t.X$ (or simply t_X) to denote the value assigned by a tuple t to the subset of variables X.

Definition 2 [Weighted Relation] A weighted relation is a pair: (r, w), where r is a relation and w is a function defined on the scheme of r. That is, for each tuple $t \in r$ there is a weight assigned whose value is $w(t)$. A weighted relation (r, w) can be represented by a single relation whose scheme is $R \cup \{W\}$ where R is the scheme of r and W is a weight variable whose value is determined by the weight function w.

Definition 3 [Bayesian Networks (Pearl, 1988)] A *Bayesian network* (BN) is a pair (G, P) where G is a directed acyclic graph over the nodes X and P are the conditional probability tables over the families of G. We represent a BN as a database with a weighted relation for each node. The weighted relation for node X_i is defined on the scheme $\{X_i\} \cup pa(X_i)$ and consists of the tuples t whose weight is the probability $P(t_{X_i} | t_{pa(X_i)})$. Only tuples having probability greater than zero are included in the relations.

Definition 4 [BN Tasks] Two of the most common BN tasks are posterior distribution computation and computation of the most probable explanation (MPE). The posterior distribution task is to compute the distribution $P(X \mid e)$ for a set of unobserved variables $X = (X_1, \ldots, X_n)$ and a set of observations, or evidence e. The MPE task is to find an assignment $x^o = (x_1^0, \ldots, x_n^o)$ such that $x^o = arg\ max_x \Pi_{i=1}^{n} P(x_i \mid x_{pa_i}, e)$.

Definition 5 [Constraint Network (Dechter, 1992)] A *constraint network* over a set of variables X consists of a set of constraints, $\{C_1, \ldots, C_t\}$. Each *constraint* C_i is a relation on a subset $S_i \subset X$ whose tuples are all the compatible value assignments. The set S_i is called the *scope* of C_i. An assignment of a unique domain value to each member of some subset of variables is called an *instantiation*. A consistent instantiation of *all* the variables that does not violate any constraint is

called a *solution*. Typical queries associated with constraint networks are to determine whether a solution exists and to find one or all solutions. The database representation of a constraint network is straightforward: each constraint is mapped into a relation. No weight attributes are necessary.

Definition 6 [Decomposable Criterion Function (Bacchus and Grove, 1995)] A criterion function, f, over a set X of n variables $X_1, ..., X_n$ having domains of values $D_1, ..., D_n$ is additively decomposable relative to a scheme $Q_1, ..., Q_t$ where $Q_i \subseteq X$ iff

$$f(x) = \sum_{i \in I} f_i(x_{Q_i}),$$

where $I = \{1, ..., n\}$ is a set of indices denoting the subsets of variables $\{Q_i\}$ and x is an instantiation of all the variables. The functions f_i are the components of the criterion function and are specified, in general, by stored tables.

Definition 7 [Constraint Optimization (Dechter *et al.*, 1988)] Given a constraint network over a set of n variables $X = X_1, ..., X_n$ and a set of constraints $C_1, ..., C_k$ having scopes $S_1, ..., S_k$, and given a criterion function f decomposable into $\{f_1, ..., f_l\}$ over $Q_1, ..., Q_l$, the constraint optimization problem is to find a consistent assignment $x = (x_1, ..., x_n)$ such that the criterion function $f = \sum_i f_i$, is maximized. Our database representation is as follows. Each function *component* f_i is represented by a weighted relation on a *scheme* Q_i with the weight being the value of f_i for every instantiation of the component variables from their domains. Each constraint is represented by a relation on the constraint's scope without need for a weight attribute. If a constraint scope coincides with a component's scheme, then we merge both the constraint and the component into a single weighted relation specified by the tuples in the constraint and the function valuation for each tuple.

Definition 8 [Influence Diagrams, Decision Networks (Howard and Matheson, 1984)] An *influence diagram* (ID) is a dag whose nodes are of three types: decision nodes, D, represented by boxes; chance nodes, C, represented by circles; and utility nodes, U, represented by diamonds. Arrows between node pairs indicate influences of two types: (a) Informational influences, represented by edges directed to a decision node; these show exactly which variables will be known by the decision maker at the time the decision is made. (b) Conditioning influences, represented by edges directed to a chance (utility) node; these show the variables on which the probability (utility) assignment to the chance (utility) node will be conditioned. Each utility node represents one component of additively decomposable utility function:

$$u(x) = \sum_{i \in I} u_i(x_{Q_i}),$$

where $I = \{1, ..., n\}$ is a set of indices denoting the utility nodes in U and the subset of variables $\{Q_i\}$ are the parents of the utility nodes and x is an instantiation of all the chance and decison variables. A *decision network* is an ID: (1) that implies a total ordering among decision nodes; (2) where each decision node and its direct predecessors (parents) directly influence all successor decision nodes. Requirement (1) is the "single decision maker" condition and requirement (2) is the "no forgetting" condition. These two conditions guarantee that a decision tree can be constructed, possibly after some probabilistic manipulation. Our database representation of an ID is as follows. Each component of the utility function u_i is represented by a weighted relation on a scheme Q_i with the weight being the value of u_i for every instantiation of the component variables from their domains. Each conditional probability for a chance node is represented by a weighted relation on the ID family with the weight being the probability.

Definition 9 [Finding the MEU, MEU policy] One main objective of inference in the influence diagram is to determine the decision policies for each decision node that maximize the overall expected utility of the influence diagram. The task of finding the MEU and the MEU policy for an ID having chance nodes C, decision nodes D and an additively decomposable utility function $u(x)$ relative to $Q_1, ..., Q_t$, $Q_i \subseteq X$, $X = C \cup D$, is defined as follows. We define a *decision function* for every decision node $X_i \in D$ as a mapping $\delta_i : x_{pa_i} \rightarrow x_i$ where x is an instantiation of all the chance and decison variables. The set of all the decision functions for a decision node X_i, denoted by Δ_i, is called the *decision function space* for X_i. The Cartesian product $\Delta = \Pi_{X_i \in D} \Delta_i$ is called the *policy space*. Given a policy $\delta = \{\delta_{X_i} \mid \forall X_i \in D\} \in \Delta$, a probability P_δ can be defined over the random nodes and the decision nodes as follows:

$$P_\delta(x) = \Pi_{X_i \in C} P(x_i | x_{pa(X_i)}) \Pi_{X_i \in D} P_{\delta_i}(x_i | x_{pa(X_i)})$$

where $P(x_i | x_{pa(X_i)})$ for $X_i \in C$ is given in the specification of the ID, while $P_{\delta_i}(x_i | x_{pa(X_i)})$ is given by δ_i as follows:

$$P_{\delta_i}(x_i | x_{pa(X_i)}) = \begin{cases} 1, & \text{if } \delta_{X_i}(x_{pa(X_i)}) = x_i \\ 0, & \text{otherwise.} \end{cases}$$

The expected utility under policy δ, denoted by $E_\delta[u]$, is defined as follows:

$$E_\delta[u] = \sum_{i \in I} u_i(x_{Q_i})\ P_\delta(x_{Q_i}),$$

The maximum of E_δ over all the possible policies is the *maximum expected utility* MEU of ID. A *MEU policy* is a a policy that achieves the MEU.

3 Structured Primitive Queries

This section defines the basic structure of the structured query language (SQL) that will be used to formulate generic primitive queries that perform basic inference operations. The primitive queries will be used in following sections to formulate compositional queries that perform inference and optimization tasks in probabilistic and deterministic networks.

3.1 SQL

SQL is a generic, non-proprietary language for the management, search, and retrieval of data. The basic structure of an SQL expression consists of **select**, **from** and **where** clauses. The **select** clause, listing attributes to be copied, corresponds to the *project* operation of relational algebra. The **from** clause, corresponding to Cartesian product, lists relations to be used. The **where** clause corresponds to selection predicate in relational algebra. A typical query has the form

$$SELECT\ A_1, A_n, \ldots, A_n$$
$$FROM\ r_1, r_2, \ldots, r_m\ WHERE\ P$$

where each A_i represents an attribute, each r_i a relation, and P is a predicate. SQL may internally convert this query into a more efficient relational algebra expression, but the semantics will be equivalent to:

$$\Pi_{A_1, A_2, \ldots, A_n}(\sigma_P(r_1 \times r_2 \times \ldots \times r_m))$$

That is, SQL forms the Cartesian product of the relations named, performs a selection using the predicate, then projects the result onto the attributes named. The result of an SQL query is a relation. SQL allows arithmetic expressions and provides aggregate operators such as **sum, min, max**. SQL uses a **group by** clause to partition the tuples in a result before application of an aggregate operator.

3.2 Primitive Queries

We present four generic SQL queries called conditioning, combination, elimination, and solution queries.

Algorithm compile-db
input: A deterministic or probabilistic network represent by a database $\Delta = \{r_1, ..., r_n\}$ of weighted relations; A set of conditioning variables O.
output: A query composition graph $G = (N, E)$; N are the nodes and E the edges.
initialization: N and E are initially empty.

1. Create a relation $assoc = \cup_{r_i \in \Delta} assoc(r_i)$ where $assoc(r_i)$ is the set of all pairs (X_j, r_i) such that X_j is a non-weight variable in the scheme of r_i.
2. **For** each relation $r_i(R_i)$ whose scheme includes conditioning variables $O_i = assoc(r_i) \cap O$, create a conditioning query $q_{cond} = cond(r_i; O_i)$; remove all associations for r_i and add those for q_{cond}: $assoc \leftarrow assoc \cup assoc(q_{cond}) \setminus assoc(r_i)$
3. Select an ordering $(X_1, X_2, \ldots, X_n)$ of all the variables in *assoc*.
4. **For** $i = n$ downto 1 **do**
   (a) **If** exist more than one relation in *assoc* that mentions X_i **then**
      i. $\Delta_i \leftarrow \{r_j \mid (X_i, r_j) \in assoc\}$
      ii. Create a combination query $q_i^c = \otimes_{\Delta_i} r_j$
      iii. Remove all associations for all $r_j \in \Delta_i$ and add those for q_i^c: $assoc \leftarrow assoc \cup assoc(q_i^c) \setminus \cup_{r_j \in \Delta_i} assoc(r_j)$
      iv. Add the node q_i^c to N and add the directed edges to E from every relation in Δ_i to q_i^c
   End if
   (b) Select the one relation r_k in *assoc* that mentions X_i
   (c) Create an eliminating query $q_i^e = \oplus_{X_i} r_k$
   (d) Remove all associations for r_k and add those for q_i^e: $assoc \leftarrow assoc \cup assoc(q_i^e) \setminus assoc(r_k)$
   (e) Add the node q_i^e to N and the directed edge (r_k, q_i^e) to E
   End for

Figure 1: Algorithm compile-db.

Each query performs a basic inference operation: (a) conditioning a relation on a subset of variables; (b) combining two or more relations on shared variables; (c) eliminating a variable from a relation; (d) solving an eliminated variable conditioned on other variables in a relation.

3.2.1 Conditioning

The query $cond(r, X)$, written $\delta_{X=x}\ r$, takes a weighted relation $r(R \cup W)$ and a subset $X \subset R$ and returns the relation obtained by selecting only tuples from r whose X value equals a parameter x and by projecting onto $R \setminus X$,

$$\delta_{X=x}\ r \stackrel{\text{def}}{=} SELECT\ r.[R \setminus X]\ FROM\ r \\ WHERE\ X = x$$

3.2.2 Combination

The query $comb(r_1, r_2)$, written $r_1 \otimes r_2$, takes two weighted relations $r_1(R_1 \cup W), r_2(R_2 \cup W)$ having shared variables $X = R_1 \cap R_2 \neq \emptyset$ and computes a relation on $R_1 \cup R_2 \cup W$ obtained by selecting tuples from both relations having same values on shared variables and by combining the weights using a combination function f,

$$r_1 \otimes r_2 \stackrel{\text{def}}{=} SELECT\ r_1.R_1, r_2.[R_2 \setminus X], \\ f(r_1.W, r_2.W) \\ FROM\ r_1, r_2\ WHERE\ r_1.X = r_2.X$$

The combining function depends on the type of weights. If the weight is of type probability, then f is a multiplication. If the weight is of type utility or cost, then f is a summation. For non-weighted relations, the combining operation is precisely the equi-join.

3.2.3 Elimination

The query $elim(r, X)$, written $\oplus_X\ r$, takes a weighted relation, $r(R \cup W)$, and a subset, $X \subset R$, and returns a new relation obtained by projecting out X and aggregating the weights. The aggregation function g is a summation for probability and a maximiztion (minimization) for utilities (costs).

$$\oplus_X\ r \stackrel{\text{def}}{=} SELECT\ r.[R \setminus X], g(W) \\ FROM\ r\ GROUP\ BY\ r.[R \setminus X]$$

3.3 Solution

The query $sol(r, X)$ takes a weighted relation, $r(R \cup W)$, and returns a relation, $x^*|_r$ that gives all solutions for X conditioned on $R \setminus X$. The SQL expression for a solution query depends on the type of the weight attribute. If the weight is a probability, then the query is the normalization query:

$$x^*|_r \stackrel{\text{def}}{=} SELECT\ r.R,\ r.W/[\oplus_X\ r].W\ As\ W \\ FROM\ \oplus_X\ r,\ r \\ WHERE\ [\oplus_X\ r].[R \setminus X] = r.[R \setminus X]$$

If the weight is a cost or a utility, then the query is the join query:

$$x^*|_r \stackrel{\text{def}}{=} SELECT\ r.R\ FROM\ \oplus_X\ r,\ r \\ WHERE\ [\oplus_X\ r].W = r.W \\ AND\ [\oplus_X\ r].[R \setminus X] = r.[R \setminus X].$$

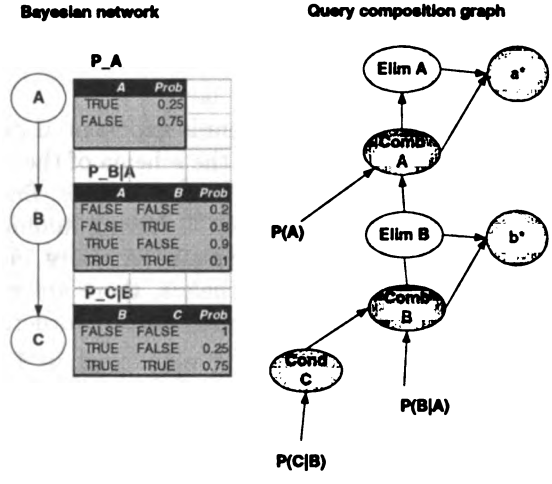


Figure 2: A Bayesian network and compiled query composition graph.

4 Query Composition and Compilation

In this section we present a compilation algorithm for composing queries from the network base relations to perform reasoning tasks in deterministic and probabilistic networks.

The algorithm computes a query composition graph (QCG) which is a directed acyclic graph (DAG) whose root nodes are the base relations and the non-root nodes are primitive queries. There is a one-to-one mapping between variables and elimination queries. Only the predecessors of each elimination query include the eliminated variable in their schema.

The QCG is computed only once and can be reused in conjunction with a spectrum of task-specific answer

queries to compute the solution for a variety of inference and optimization tasks.

Figure 1 gives a high-level description of the compilation algorithm, called *compile_db*. The algorithm takes the network database schema, and an ordering of the network variables, and outputs a QCG obtained by eliminating the variables one by one starting with the last variable in the ordering. When variable i is eliminated, all relations that mention the variable are collected. If there is more than one relation in the collection, then the relations are first combined in one relation, using a combination query, and then the variable i is eliminated using an elimination query. If there is only one relation that mentions i, then the elimination query for i uses that relation. The algorithm maintains an association table between variables and relations and as queries are computed the dependencies are recorded as the directed edges in the QCG. The elimination process continues until all variables are eliminated

The complexity of the algorithm is exponential in the size of the maximum clique or the schema of the combination queries. As in computing join trees, finding an ordering that produces a QCG having minimum-size largest clique is NP-complete (Arnborg, 1985; Arnborg *et al.*, 1987). Nevertheless, there are many greedy algorithms that can identify effective orderings.

Example 1 (Posterior distribution) Figure 2 shows a Bayesian network's acyclic graph and the relations encoding the conditional probability tables. The Figure also shows the QCG for an evidence on C and using the elimination order: (C, B, A). The steps of compile_db and the computed queries are as follows:

1. The set of association pairs between variables and relations is: $assoc \leftarrow \{(C, P_{C|B}), (B, P_{C|B}), (B, P_{B|A}), (A, P_{B|A}), (A, P_A)\}$

2. The conditioning variable is C. There is only one relation on C in *assoc*. The algorithm computes the conditioning query *cond_C*. The query conditions the relation $P_{C|B}$ and is given by a query parameterized on the evidence c:

```
PARAMETERS [c] Bit;
SELECT [P_C|B].B, [P_C|B].prob
FROM [P_C|B]
WHERE ((([P_C|B].C)=[c]));
```

3. $assoc \leftarrow \{(B, cond\_C), (B, P_{B|A}), (A, P_{B|A}), (A, P_A)\}$

4. To eliminate B, determine all relations associated with B. The relations are: $cond\_C, P_{B|A}$. The relations need to be combined by a combination query, *comb_B*, and is given by:

```
SELECT cond_C.B, [P_B|A].A,
 cond_C.prob*[P_B|A].prob AS prob
FROM cond_C, [P_B|A]
WHERE cond_C.B =  [P_B|A].B;
```

5. $assoc \leftarrow \{(B, comb\_B), (A, comb\_B), (A, P_A)\}$

6. compile_db eliminates B from the relation *comb_B*. *elim_B* is a marginalization query and is given by:

```
SELECT comb_B.A, Sum(comb_B.prob) AS prob
FROM comb_B
GROUP BY comb_B.A;
```

7. $assoc \leftarrow \{(A, elim\_B), (A, P_A)\}$

8. Compute the solution query for the eliminated variable. b^* is a solution query that computes the posterior of B conditioned on A and is given by:

```
SELECT comb_B.B, comb_B.A,
 comb_B.prob/elim_B.prob as prob
FROM elim_B, comb_B
WHERE elim_B.A = comb_B.A;
```

9. To eliminate A, find all relations associated with A. The relations are: $elim\_B, P_A$. The relations need to be combined by a combination query, *comb_A*, given by:

```
SELECT P_A.A,
 P_A.prob * elim_B.prob as prob
FROM elim_B, P_A
WHERE  P_A.A =  elim_B.A;
```

10. $assoc \leftarrow \{(A, comb\_A)\}$

11. Eliminate A by the query *elim_A* given by:

```
SELECT Sum(comb_A.prob) AS prob
FROM comb_A;
```

12. $assoc \leftarrow \emptyset$

13. Compute the solution query a^*, which is the posterior probability of A, given by:

```
SELECT comb_A.A,
  comb_A.prob/elim_A.prob AS prob
FROM elim_A, comb_A;
```

Executing the query procedure on the evidence $c = True$ produces the solutions:

b^*		
B	A	$prob$
True	True	1
True	False	1

a^*	
A	$prob$
True	0.04
False	0.96

Example 2 (MPE) Again, consider the network of Figure 2. The QCG structure is as shown in the Figure. The queries are identical to those for posterior beliefs, except for the elimination and solution queries. The aggregation function in the elimination query is a maximization, instead of summation. For example, the query $elim\_B$ becomes

```
SELECT comb_B.A, Max(comb_B.prob) AS prob
FROM comb_B
GROUP BY comb_B.A;
```

The solution query for the MPE are join queries, as in optimization problems. For example, the solution query b^* becomes

```
SELECT comb_B.B, comb_B.A
FROM elim_B, comb_B
WHERE elim_B.A = comb_B.A
AND comb_B.prob = elim_B.prob;
```

Executing the queries on the evidence $c = True$ we get the following solution tables

b^*	
B	A
True	True
True	False

a^*
A
False

5 Answer Queries

Given a query composition graph, we can efficiently answer queries on various subsets of variables. These answer queries are computed from the composition graph in linear time starting from the node of the last eliminated variable.

Figure 3 describes the algorithm AQ for computing the answer queries on a set of query variables. The algorithm first combines the solution tables sequenetially in reverse elimination ordering, until all query variables are included, then returns the projection on the query variables. For optimization problems, the solution relations are non-weighted relations and the combination operation becomes the inner join.

Example 3 Consider the network of Figure 2. The answer query for the posterior probability of (A, B) given the evidence $C = c$ is obtained by combining the tables for a^*, b^* given in Example 1. For the evidence $c = True$ we get

A	B	$prob$
True	True	0.04
False	True	0.96

The answer query for the MPE is obtained by joining the tables a^* and b^* for the evidence $C = c$. Based on the tables in Example 2 the MPE for the evidence $c = True$ is: $(a^* = False, b^* = True)$.

Algorithm Answer-query (AQ)
input: Query variables Q and Query composition graph G for elimination ordering $(X_n, \ldots, X_1)$
output: All solutions for Q
1. $q_A \leftarrow \emptyset$
2. $i = 0$
3. Repeat until $scheme(q_A) \supseteq Q$
4. i=i+1
5. $q_A \leftarrow q_A \otimes x_i^*$
6. Loop
7. Return the projection $\pi_Q\ q_A$

Figure 3: Algorithm answer-query (AQ).

6 Finding the MEU

In this section we describe the QCG compilation procedure for finding the MEU and computing the MEU policy. We consider influence diagrams representing a single decision-maker, who does not forget the information. That is, the direct predecessor set for one decision must be a subset of the direct predecessor set of any subsequent decision.

Decisions are eliminated forming a join on the decision and its information predecessors. The information predecessors $I(D)$ of a decision D, is the set of variables that are observed prior to making the decision. Thus, if decision D has information predecessors I, then the best decision for D is a function $D^*(I)$. Decisions are totally ordered and satisify the no forgetting property: if decision D_1 preceeds D_2 then the information predecessors of D_1 are also information predecessors of D_2 (that is, $I(D_1)$ is a subset of $I(D_2)$[2]).

If decision nodes have parents that are probability variables, then maximization steps must be interleaved

[2]The set of required information predecessors for a decision node does not need to include the information predecessors of all preceeding decision nodes in some circumstances. See (Shachter, 1998), for details.

with summation steps. Let I_{i+1} be the set of information predecessors that will be observed between decision D_i and D_{i+1}. I_0 is the set of information predecessors that will be observed prior to D_1. I_n is the set of uncertainties that will never be observed prior to any decision.

Given the ordering $(D_1, \ldots, D_n)$ on the decisions in D, the expected utility of the decision problem is

$$E[u] = \sum_{I_0} \max_{D_1} \sum_{I_1} \max_{D_2} \ldots \sum_{I_{n-1}} \max_{D_n} \sum_{I_n} P(X)u(X)$$

The decision policies D_i^* are the functions that maximize this expression.

When constructing the QCG, uncertainties and decisions must be eliminated in a more constrained order than in a QCG for probabilities. The decisions D and their information predecessors induce a partial order on the nodes in X: $I_0 \prec D_1 \prec I_1 \prec \ldots \prec D_n \prec I_n$. This partial order dictates the *required* order of elimination; that is, the variables in I_k must be eliminated before the variables in $I_0 \cup D_1 \cup I_1 \cup \ldots \cup I_{k-1} \cup D_k$. The elimination sequence is the *reverse* of some total order that is an extension of the partial order $I_0 \prec D_1 \prec I_1 \prec \ldots \prec D_n \prec I_n$ A decision must be joined to all of its information predecessors before eliminating the decision.

Our algorithm follows (Jensen *et al.*, 1994). Decisions are eliminated after forming a join on the decision a join on the decision and its information predecessors, $I(D_k) = I_0 \cup D_1 \cup I_1 \cup \ldots \cup D_{k-1} \cup I_{k-1}$, and selecting the decision value for each combination of information predecessors that maximizes the utilty.

The weighted relations for decision problems can have both utility and probability weights. The default weight for probability is 1 and for utility is 0. If a relation does not specify a value for the weight attribute (utility or probability), then the default value is considered. The definitions for the querying operations (combination, elimination, and solution) are extended as follows:

Combination The combination of two weighted relations $r_1(R_1 \cup \{Util, Prob\}), r_2(R_2 \cup \{Util, Prob\})$ having shared variables $X = R_1 \cap R_2 \neq \emptyset$ is a relation on $R_1 \cup R_2 \cup \{Util, Prob\}$ obtained by the query:

$$\begin{aligned} r_1 \otimes r_2 \stackrel{\text{def}}{=}\ & SELECT\ r_1.R_1, r_2.[R_2 \setminus X], \\ & (r_1.Prob \times r_2.Prob)\ As\ Prob, \\ & (r_1.Util/r_1.Prob + r_2.Util/r_2.Prob)\ As\ Util \\ & FROM\ r_1, r_2\ WHERE\ r_1.X = r_2.X \end{aligned}$$

Elimination The elimination query depends on the type of eliminated node. Eliminating a chance node is done by summing out probabilities and taking expectation of utilities:

$$\begin{aligned} \oplus_X r \stackrel{\text{def}}{=}\ & SELECT\ r.[R \setminus X], \\ & Sum(r.Util \times r.Prob)\ AS\ Util, \\ & Sum(r.Prob)\ AS\ Prob \\ & FROM\ r\ GROUP\ BY\ r.[R \setminus X] \end{aligned}$$

Eliminating a decision node is obtained by maximizing utilities:

$$\begin{aligned} \oplus_X r \stackrel{\text{def}}{=}\ & SELECT\ r.[R \setminus X], r.Prob\ AS\ Prob \\ & Max(r.Util)\ AS\ Util, \\ & FROM\ r \\ & GROUP\ BY\ r.[R \setminus X], r.Prob \end{aligned}$$

Solution The solution query also depends on the type of solved node. Solving for a chance node is done by a normalization query:

$$\begin{aligned} x^* |_r \stackrel{\text{def}}{=}\ & SELECT\ r.R, \\ & r.Prob/[\oplus_X r].Prob\ As\ Prob \\ & FROM\ \oplus_X r, r \\ & WHERE\ [\oplus_X r].[R \setminus X] = r.[R \setminus X] \end{aligned}$$

Solving for a decision node is done by a join query:

$$\begin{aligned} x^* |_r \stackrel{\text{def}}{=}\ & SELECT\ r.R, r.Util \\ & FROM\ \oplus_X r, r \\ & WHERE\ [\oplus_X r].Util = r.Util \\ & AND\ [\oplus_X r].[R \setminus X] = r.[R \setminus X] \end{aligned}$$

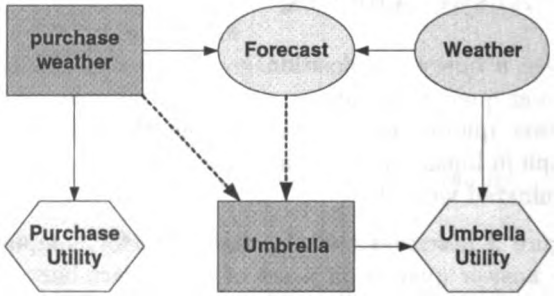


Figure 4: Influence diagram for the umbrella example

Example 4 Consider the umbrella decision problem depicted by the influence diagram in Figure 4. The dotted edges in the Figure represent informational influences. The problem consists of a sequential decision problem where the first decision is whether or

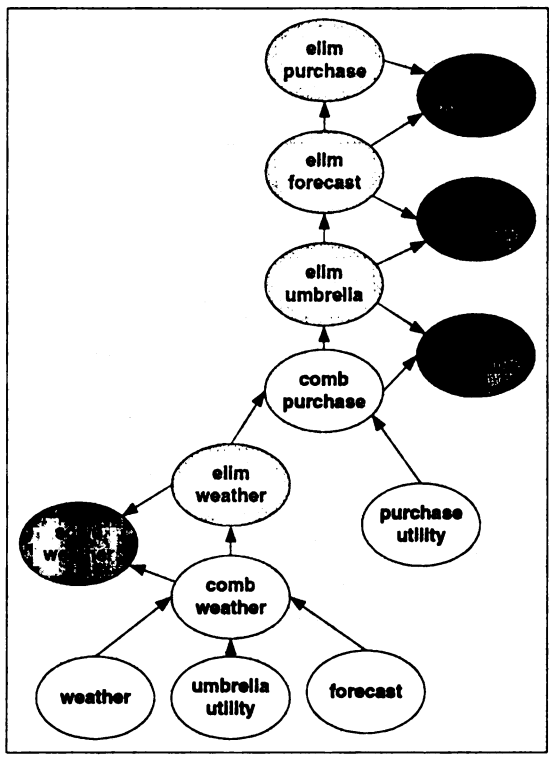


Figure 5: QCG for the umbrella example

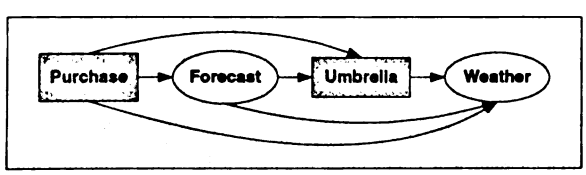


Figure 6: Solution dag for the umbrella example

not to purchase weather information and the second is whether or not to take the umbrella. The umbrella decision is contingent on weather forecast information if we opt to purchase the forecast. The decisions and their information predecessors induce the partial order: *purchase* $\prec$ *forecast* $\prec$ *umbrella* $\prec$ *weather*, which is also a total order. The elimination order is the reverse ordering: *(weather, umbrella, forecast, purchase)*. The database encoding consists of four weighted relations: two for the chance nodes (weather and forecast) and two for the utility components (u_umbrella and u_purchase). The following is a quantification of those relations:

Weather	
Weather	Prob
sun	0.6
rain	0.4

Forecast			
Purchace	Weather	Forecast	Prob
yes	sun	sunny	0.7
yes	sun	cloudy	0.2
yes	sun	rainy	0.1
yes	rain	sunny	0.15
yes	rain	cloudy	0.25
yes	rain	rainy	0.6
no	sun	none	1
no	rain	none	1

u_umbrella		
Weather	Umbrella	Util
rain	take	70
rain	leave	0
sun	take	20
sun	leave	100

u_purchase	
Purchase	Util
yes	-5
no	0

The QCG compiled for the example is shown in Figure 5. Algorithm compile_db proceeds as follows. First we record in *assoc* all associations between variables and relations. To eliminate *weather* we remove all entries associated with the variable *weather* in *assoc* which are *weather, forecast, u_umbrella*. We combine these relations using a combination query. The relation returned by the combination query will have two weight columns: one is a probability which is the product of the probabilities in *weather* and *forecast*, and the other is the utility from *u_umbrella*. When combining tables with different types of weights, as in this example, use a weight of 0 for a missing utility columm and 1

for a missing probability column. We then update *assoc* by adding entries of *comb_weather* associated with the clique: *Weather, Umbrella, Forecast, Purchace.* The next step is to compute an elimination query for weather which projects out the weather column from *comb_weather*, summing out probabilites and taking expectation of utilities. We update the association table by removing all entries for *comb_weather* and add those for *elim_weather*. We then compute a solution query for *weather* which is a normalization query for the probability of weather conditioned on the remaining variables. The result for the query is:

solve_weather				
Umbrella	Forecast	Purchase	Weather	Prob
leave	cloudy	yes	sun	0.545
leave	cloudy	yes	rain	0.455
leave	none	no	sun	0.6
leave	none	no	rain	0.4
leave	rainy	yes	rain	0.8
leave	rainy	yes	sun	0.2
leave	sunny	yes	rain	0.125
leave	sunny	yes	sun	0.875
take	cloudy	yes	rain	0.455
take	cloudy	yes	sun	0.545
take	none	no	sun	0.6
take	none	no	rain	0.4
take	rainy	yes	sun	0.2
take	rainy	yes	rain	0.8
take	sunny	yes	sun	0.875
take	sunny	yes	rain	0.125

Althought the umbrella decision is part of the solution schema the probability will be conditionally independent of the umbrella decision given the forecast and the purchase decision. Next, to eliminate the decision node *umbrella* we remove from *assoc* all relations that mention that node or any of its information predecessors. We get the two relations *elim_weather* and *u_purchase*. The relations are then combined by a combination query which computes the following relation:

comb_purchase				
Umbrella	Forecast	Purchase	Prob	Util
leave	cloudy	yes	0.22	49.54545
leave	none	no	1	60
leave	rainy	yes	0.3	15
leave	sunny	yes	0.48	82.5
take	cloudy	yes	0.22	37.7273
take	none	no	1	40
take	rainy	yes	0.3	55
take	sunny	yes	0.48	21.25

The elimination query for eliminating *umbrella* from *comb_purchase* is computed by instantiating the generic elimination query for decision node, which computes the following relation:

elim_umbrella			
Forecast	Purchase	Prob	Util
cloudy	yes	0.22	49.54545
none	no	1	60
rainy	yes	0.3	55
sunny	yes	0.48	82.5

The elimination process continues until all variables are eliminated. The solution queries for the remaining nodes are:

solve_umbrella			
Umbrella	Forecast	Purchase	Utility
leave	cloudy	yes	49.545
leave	none	no	60
leave	sunny	yes	82.5
take	rainy	yes	55

solve_forecast		
Purchase	Forecast	probability
no	none	1
yes	cloudy	0.22
yes	rainy	0.3
yes	sunny	0.48

solve_purchase	
Purchase	Utility
yes	67

The final elimination query will return the MEU and an answer query will consist of the MEU policy. The schemes of the solution queries constitute a solution dag, shown in Figure 6, specifying the policy for each decision node as function of its information predecessors. The MEU policy is constructed by joining the solution tables for the decision nodes. The policy is: (1) purchase weather information; (2) take umbrella if forecast is rain and leave otherwise.

7 Discussion and Related Work

Our database encoding of Bayesian networks adopts a form of zero compression (Jensen and Andersen, 1990; Huang and Darwiche, 1994) which avoids recording tuples having zero probability. Our approach differs from previous work in both generality and scope and focuses on a unifying compositional approach to inference and optimization tasks in deterministic and probabilistic networks. Our algorithms for compiling the QCG and for computing answer queries are equivalent to to other

vertex elimination algorithms (Zhang and Poole, 1994; Dechter, 1996). There are two essential differences between our approach and previous elimination algorithms. One, our QCG enables using a single elimination process to reason about diverse queries by only modifying generic combination and aggregagtion functions. Two, given a QCG we can instantiate generic answer queries to perform a spectrum of inference and optimization tasks. All the queries are expressed in relational algebra using SQL and can be implemented on standard database query engines. Our work is similar to that of (Wong *et al.*, 1995) in that it also uses SQL queries for probabilistic inference. Our algorithm for the MPE task is similar to that of (Seroussi and Golmard, 1994). Various unifying approaches observing the common features between various algorithms were also proposed by (Shenoy, 1992). A closely related approach to optimization in constraint networks (Dechter *et al.*, 1990) also uses solution dependency structure similar to ours. Our approach enables a hybrid representation combining categorical constraints and conditional probability tables which can be paricularly useful in the context of decision problems and influence diagrams.

8 Concluding Remarks

In this paper we have presented a compositional approach to automated inference based on a unifying database representation. We provided an elimination algorithm for compiling a database schema representation of a probabilistic or deterministic network to a query composition graph (QCG). The root nodes of the graph are the network base relations and the nodes are instantiations of generic primitive queries. Each query in the graph represents a view or a derived relation from the network stored base relation. The QCG can be used for performing diverse inference and optimization tasks by computing answer queries which can be done in linear time. Besides generality, an advantage of our compositional approach is that it enables incremental compilation and updates. Incremental compilation can be attained by compiling the database incrementally; at each increment the compiled portion of the database is replaced by the QCG whose queries represent "materialized views" that are cached with the uncompiled portion of the database. Updating the QCG to insert or retract evidence or to condition on a subset of variables can be done incrementally by local propagation on the QCG.

References

S.A. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal of Discrete Mathematics*, 8:277–284, 1987.

S.A. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability - a survey. *BIT*, 25:2–23, 1985.

F. Bacchus and A Grove. Graphical models for preferences and utility. In *Uncertainty in AI (UAI-95)*, pages 3–10, 1995.

R. Dechter, A. Dechter, and J. Pearl. Optimization in constraint networks. In R. M. Oliver and J. Q. Smith, editors, *Influence Diagrams Belief Nets and Decision Analysis*, pages 411–425. Wiley, 1988.

R. Dechter, A. Dechter, and J. Pearl. Optimization in constraint networks. In R.M. Olivier and J.Q. Smith, editors, *Influence Diagrams, Belief Nets and Decision Analysis*, pages 411–425. J. Wiley, New York, 1990.

R. Dechter. Constraint networks. *Encyclopedia of Artificial Intelligence*, pages 276–285, 1992.

R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In E. Horvitz and F. Jensen, editors, *Uncertainty in Artificial Intelligence*, pages 211–219. Morgan Kaufmann, 1996.

Y. El Fattah. Structured modeling language for automated modeling in causal networks. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 1108–1114, Stockholm, Sweden, 1999.

R. A. Howard and J. E. Matheson. Influence diagrams. In R. A. Howard and J. E. Matheson, editors, *The Principles and Applications of Decision Analysis, Vol. 2.* Strategic Decision Group, Menlo Park, CA, 1984.

C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 11, 1994.

F. Jensen and S.K. Andersen. Approximations in bayesian belief universes for knowledge-based systems. In *Uncertainty in Artificial Intelligence (UAI-90)*, pages 162–169, 1990.

F. Jensen, F. V. Jensen, and S. L. Dittmer. From influence diagrams to junction trees. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pages 367–373, 1994.

D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 580–587, 1998.

D. Maier. *The Theory of Relational Databases.* Computer Science Press, Rockville, MD, 1983.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, 1988.

B. Seroussi and J. L. Golmard. An algorithm directly finding the k most probable configurations in bayesian networks. *International Journal of Approximate Reasoning*, 11:205–233, 1994.

Ross D. Shachter. Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, 1998.

P. P. Shenoy. Valuation based systems for bayesian decision analysis. *Operations Research*, 40:463–484, 1992.

W. X. Wen. From relational databases to belief networks. In *Uncertainty in Artificial Intelligence (UAI-91)*, pages 406–413, 1991.

S.K.M. Wong, C.J. Butz, and Y. Xiang. A method for implementing a probabilistic model as a relational database. In *Uncertainty in Artificial Intelligence (UAI-95)*, pages 556–564, 1995.

N. L. Zhang and D. Poole. A simple approach to bayesian networks. In *Proceedings of the Tenth Biennial Canadian Artificial Intelligence Conference on Uncertainty in Artificial Intelligence (AI-94)*, pages 171–178, 1994.

An Alternative Combination of Bayesian Networks and Description Logics

Phillip M. Yelland
Sun Microsystems Laboratories
901 San Antonio Road, MS MTV29-117
Palo Alto, CA 94303-4900.
phillip.yelland@sun.com

Abstract

A number of knowledge representation formalisms have been proposed that seek to combine the capabilities of description logics with facilities for probabilistic inference. One of these—P-CLASSIC—extends the widely-used description logic CLASSIC with Bayesian networks, a popular framework for reasoning with probabilities. To fashion such a union, however, the authors of P-CLASSIC were compelled to place restrictions on the kinds of assertion that could be made in the system. The system outlined in this paper also blends description logics and Bayesian networks; unlike P-CLASSIC, however, it is obtained by admixing the facilities of description logics into Bayesian networks, rather than vice versa. The result is less expressive than P-CLASSIC in certain respects, but it is able to capture aspects of knowledge that are beyond P-CLASSIC.

1 INTRODUCTION

The last decade or so has witnessed the ascendancy of two very different knowledge representation formalisms: *Bayesian networks* (*BN*'s) are primarily graphical devices that facilitate the concise and tractable expression of probabilistic knowledge. *Description logics* (*DL*'s) are subsets of first-order logic that may be used to describe complex concepts.

In many ways, the capabilities of BN's and DL's are complementary. For BN's typically encompass propositional assertions only, and are incapable of capturing rich concept descriptions. Conversely, most DL's have very limited facilities for modeling uncertainty. One might consider, therefore, combining the two formalisms, so as to compound their strengths and ameliorate their respective weaknesses. The P-CLASSIC system (Koller, Levy and Pfeffer 1997) seeks to do just that. An elaboration of the successful DL CLASSIC, P-CLASSIC provides an expansive framework for concept description, fortified with Bayesian network facilities to express probabilistic knowledge about such concepts. Unfortunately, in order to circumvent some of the difficulties highlighted in subsequent sections of this paper, the authors of P-CLASSIC were compelled to place restrictions on the kind of assertions that could be expressed in the system (see Section 8.1 for details). Even in the presence of such restrictions, P-CLASSIC is undoubtedly useful (Mantay and Moller 1998), but for some applications—such as that described later in this paper—they can cause significant difficulties.

This paper describes a knowledge representation formalism that, like P-CLASSIC, blends description logics and Bayesian networks. Unlike P-CLASSIC, however, it does not have its provenance in an existing description logic. This allows the expressive power of the framework's description logic component to be deliberately circumscribed, making it possible to abrogate many of the restrictions required by P-CLASSIC.

The paper begins with brief descriptions of Bayesian networks and description logics,[1] and of issues surrounding their combination. The combined system is presented, along with an account of its formal characteristics and a sound inference procedure for the system. A brief sketch is provided of a sample application of the system. Related existing work—in particular, P-CLASSIC—is outlined, and compared with that described here. The paper concludes by suggesting possible courses for future development.

[1] Section 2 may be skipped by those already acquainted the two formalisms.

2 BACKGROUND

2.1 BAYESIAN NETWORKS

A Bayesian network (or *BN*) is a directed acyclic graph (*DAG*) whose nodes represent random variables and whose arcs portray dependency relationships between variables. Each node in a BN is associated with a conditional probability distribution, which gives the probabilities associated with values of the variable represented by the node for different configurations of its parents' variables. A BN encapsulates a set of independence assertions between the variables it contains; each variable in the network is independent of its non-descendants given its parents.

In essence, a BN is a concise encoding of a joint probability distribution for the variables it contains. This probability distribution may be obtained—as is shown in (Jensen 1996), for example—simply by multiplying factors comprising the conditional probability distributions of each node:[2]

$$p(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} p(x_i \mid \pi_i)$$

The probability distribution of arbitrary sets of variables may be derived from the joint probability distribution by *marginalization*; to eliminate variable x_j, for instance, which ranges over the values $\{v_1, \ldots, v_n\}$, we have:

$$p(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n) = \sum_{v \in \{v_1, \ldots, v_n\}}^{n} p(x_1, \ldots, x_{j-1}, x_j = v, x_{j+1}, \ldots, x_n)$$

For BN's of any significant size, computing probabilities directly in this fashion is impractical; it requires the enumeration of sets of values whose sizes are potentially exponential in the number of variables in the network. Fortunately, researchers have devised a number of algorithms for computing probabilities in Bayesian networks that exploit their encoded independence assertions, thereby offering acceptable performance in most practical cases. Introductions to such algorithms may be found in (Castillo, Gutiérrez and Hadi 1997), (Zhang and Poole 1996) or (Jensen 1996). The framework presented in this paper does not rely on the particular characteristics of any of these algorithms, and any of them may be selected to perform the required calculations.

[2] The treatment in this paper follows the common practice of restricting BN's to (finitely-bounded) discrete random variables. The probability that random variable x takes on the value c is denoted $p(x = c)$, and $p(x)$ represents the probability distribution comprising all possible values of x. If y is another random variable, the corresponding joint and conditional probabilities are denoted $p(x = c, y = c')$, $p(x, y)$, $p(x = c \mid y = c')$ and $p(x \mid y)$. For each i, π_i is the set of parents of node x_i in the graph. In this section, it is assumed that the sequence x_1, x_2, ..., x_n is an enumeration such that each node appears after its parents.

2.2 DESCRIPTION LOGICS

Description logics grew out of attempts to formalize the *semantic networks* and *frame systems* used in many early knowledge representation efforts (Bachman and Schmolze 1985). The past twenty years have seen the development of an impressive array of description logics (MacGregor 1991), ranging in expressive power from those comparable to first-order predicate logic to far more restricted systems like those described below.

All description logics comprise a means for describing sets of objects (*concepts*) in a domain and relations (known as *roles*) between members of those concepts. By way of an example, consider the simple description logic $\mathcal{FL}^-$ introduced in (Brachman and Levesque 1984). The syntax of $\mathcal{FL}^-$ is as follows:

P	–	Primitive concept name	
R	–	Role name	
C	::=	$P \mid$	– Primitive concept
		$C_1 \sqcap C_2 \mid$	– Conjunction
		$\forall R.C \mid$	– Universal role quantification
		$\exists R$	– Existential role quantification

Primitive concepts are sets of objects that are assumed to be given. Also assumed given are roles, relating objects in the domain. Three operations are provided for defining complex concepts:

- *Conjunction:* The concept $C_1 \sqcap C_2$ comprises those objects that are members of both C_1 and C_2.
- *Universal role quantification:* All objects in the concept $\forall R.C$ are related by role R only to objects in C. The objects related to a given object by role R are known as the R-fillers of that object; thus the concept $\forall R.C$ may be described as the set of objects with R-fillers all in C.
- *Existential role quantification:* All objects in $\exists R$ have at least one R-filler.

These informal specifications can be formalized by providing a model-theoretic semantics for concept expressions. To wit, a model, $\langle D, c, r, [\_] \rangle$, for $\mathcal{FL}^-$ specifies a domain of objects D, for each primitive concept P a set $c(P) \subseteq D$, for each role R a relation $r(R) \subseteq D \times D$, and an interpretation function, $[\_]: C \rightarrow \wp(D)$, mapping concept expressions to subsets of the domain, satisfying:

$$[\![P]\!] = c(P)$$
$$[\![C_1 \sqcap C_2]\!] = [\![C_1]\!] \cap [\![C_2]\!]$$
$$[\![\forall R.C]\!] = \{o \in D \mid \forall o' \in D.\langle o, o' \rangle \in r(R) \Rightarrow o' \in [\![C_2]\!]\}$$
$$[\![\exists R]\!] = \{o \in D \mid \exists o' \in D.\langle o, o' \rangle \in r(R)\}$$

3 ISSUES INVOLVED IN COMBINING BN'S AND DL'S

Combining the two formalisms described in the previous section requires some means of adapting BN's (which normally express relationships between random variables) to the specification of relationships between sets (the denotations of DL expressions). To illustrate how this might be done, consider a BN node specifying the probability distribution of the random variable, x, with values $\{v_1,\ldots,v_n\}$. In formal terms (Halmos 1974), x defines a function f_x from a *sample space* (the set of events under consideration) to the set $\{v_1,\ldots,v_n\}$. By extension, it also defines a collection of sets $f_x^{-1}(v_1),\ldots,f_x^{-1}(v_n)$—the pre-images of each value. Since each point in the sample space is associated with precisely one value of x, this collection of pre-images partitions the space—all are disjoint, and their union equals the space itself. An equivalent formulation of the node in the network, therefore, is as an association of the pre-images $f_x^{-1}(v_1),\ldots,$ $f_x^{-1}(v_n)$ with a set of probabilities (the original probability distribution function of x). These pre-image sets may be specified directly using expressions in a description logic, rather than indirectly using values of a random variable. Each event in the sample space would be interpreted as a random selection from the domain of objects, and the probability distribution defined by the node would define the probabilities that such a random selection resulted in an object in the sets defined by the DL expressions.

Assigning DL expressions to the nodes of a BN, however, is fraught with difficulties. Consider, for example, ascribing the expression $C_1 \sqcap C_2$ to a node as part of its label set. In lieu of any further information, this would only specify the probability of the combined expression, and would fail to determine precisely the probability of either component. Such imprecision seems contrary to the spirit of the Bayesian network paradigm, which looks to produce a complete specification of the probabilities in a problem. Such an expression would also interact with labels on other nodes; for example, if C_1 occurs in the label set of another node, the drafter of the network would need to ensure that $p(C_1 \sqcap C_2, C_1) = p(C_1, C_2)$.

A possible remedy for such difficulties would involve restricting the labeling of nodes to some class of "atomic" concept descriptions that do not feature conjunctions. Candidates from the language $\mathcal{FL}^-$ would be primitive concepts, universal role quantification of primitive concepts (of the form $\forall R_1.\ldots\forall R_n.P$, say), and existential role qualification ($\exists R$). Even with such restrictions, however, problems remain. For example, if both of the expressions $\forall R.C_1$ and $\forall R.C_2$ appear in the label set of any node, then the network must assert (directly or indirectly) that $p(\exists R) = 1$; otherwise it follows that there are objects with no role fillers for R, implying that the sets $\forall R.C_1$ and $\forall R.C_2$ are not disjoint—a requirement if they are to label the same node.

Proliferation of such implicit consistency requirements can make the design of labeled networks inordinately onerous. As these examples illustrate, the interactions between different parts of such a network can be subtle, and apparently minor adjustments in one node can have ramifications throughout the network. It would be desirable to contrive a framework within which a complete probability distribution can be defined for the concepts of interest, while minimizing the possibility for inconsistencies like those above. The next section lays out such a framework.

4 TDL: A TINY DESCRIPTION LOGIC

The first step in constructing the combined framework is to adapt the description logic $\mathcal{FL}^-$. The resulting notation resembles the *logics of feature structures* (Moss 1991) employed in linguistics, in as far as it conflates universal and existential role quantification. It also invokes some of the flavor of type systems that incorporate record types and intersection (Pierce 1991).

4.1 SYNTAX

The syntax of this "Tiny Description Logic" (*TDL*) is as follows:

P	– Primitive concept name		
R	– Role name		
C	::=	$P \mid$	– Primitive concept
		$C_1 \sqcap C_2 \mid$	– Conjunction
		$R{:}C \mid$	– Role quantification

Some notation: If $\boldsymbol{S}$ is a set[3] with members C_1, C_2, ..., C_n, the expression $\bigsqcap \boldsymbol{S}$ denotes the concept $C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$. (That the latter is unambiguous follows from the commutativity and associativity of conjunction.) The expression $C \in C'$ is true iff C' can be rearranged (using commutativity and associativity) into the form $C_1 \sqcap \ldots \sqcap C_n$, where $C \in \{C_1,\ldots,C_n\}$.

[3] All variables referring to sets appear in bold font.

The *atomic expressions*, A, of TDL are characterized by the following (recursive) definition:

$$A \quad ::= \quad P \mid R{:}A$$

In the sequel, it will be convenient to assume that the set of atomic expressions is finite.[4] This restriction could be removed in principle, but the technicalities required to deal with infinite sets and probability measures over them would complicate the presentation.

4.2 SEMANTICS

The semantics of TDL resembles that of $\mathcal{FL}^-$, except that role quantification implies the existence of at least one role filler. More explicitly, a *model*, $\langle D, c, r, [\_] \rangle$, for TDL specifies a domain of objects D, for each primitive concept P a set $c(P) \subseteq D$, for each role R a relation $r(R) \subseteq D \times D$, and an interpretation function mapping concept expressions to subsets of the domain, $[\_] : C \to \wp(D)$, satisfying:

$$[\![P]\!] = c(P)$$

$$[\![C_1 \sqcap C_2]\!] = [\![C_1]\!] \cap [\![C_2]\!]$$

$$[\![R{:}C]\!] = \left\{ o \in D \;\middle|\; \begin{array}{l} \forall o'.\langle o, o' \rangle \in r(R) \Rightarrow o' \in [\![C_2]\!] \text{ and} \\ \exists o'.\langle o, o' \rangle \in r(R) \end{array} \right\}$$

A useful property of atomic expressions in TDL is encapsulated in the following result:

Proposition 1. Every concept expression C in TDL is equivalent to a conjunction $\nu(C)$ of atomic expressions.[5]

5 LABELED BAYESIAN NETWORKS

The second element of the combined framework provides a means of expressing probabilistic relationships between TDL concepts using Bayesian networks. The syntactic structure of these networks is specified below, and a guide to their interpretation follows.

5.1 SYNTAX

Definition 1. A *labeled Bayesian network* (*LBN*) $\langle n, \pi, \boldsymbol{L}, p \rangle$ comprises:

1. A DAG with nodes 1, ..., n, such that each node i has parents $\pi_i = \{\pi_{i1}, \ldots, \pi_{im}\} \subseteq \{1, \ldots, i-1\}$.
2. An indexed collection of disjoint label sets $\boldsymbol{L}_1, \ldots, \boldsymbol{L}_n$ such that each $\boldsymbol{L}_i \subseteq \boldsymbol{A}$, where $\boldsymbol{A}$ is the set of atomic TDL expressions. For convenience's sake, let $C \in \boldsymbol{L}$ abbreviate $\exists i \in \{1, \ldots, n\}. C \in \boldsymbol{L}_i$.
3. For each node i, $L \in \boldsymbol{L}_i$ and $(L_1, \ldots, L_m) \in \boldsymbol{L}_{\pi_{i1}} \times \ldots \times \boldsymbol{L}_{\pi_{im}}$, a conditional probability $p(L \mid L_1, \ldots, L_m)$.

   For all $L_1 \in \boldsymbol{L}_1, \ldots, L_n \in \boldsymbol{L}_n$, $j \in \{1, \ldots, n\}$, $i \in \{1, \ldots, j\}$, define:

$$p(L_1, \ldots, L_n) = \prod_{i=1}^{n} p(L_i \mid L_{\pi_{i1}}, \ldots, L_{\pi_{im}})$$

$$p(L_1, \ldots, L_{i-1}, L_{i+1}, \ldots, L_n) = \sum_{L \in \boldsymbol{L}_i} p(L_1, \ldots, L_{i-1}, L, L_{i+1}, \ldots, L_n)$$

Each node of a labeled Bayesian network represents a collection of disjoint sets of objects denoted by atomic expressions in TDL.[6] The conditional probabilities in such a network assert the probability that an object selected at random will be a member of a given set, given that it is a member of certain other sets.

That LBN's express relationships between atomic concepts only is less of a restriction than it might appear, given that, as shown in Proposition 1, every concept in TDL can be represented as a conjunction of atomic concepts. The definition below specifies how such a conjunction can be construed as a "joint event"—the event that a randomly selected object is a member of all the sets in the conjunction. Full details of the treatment of concepts in general are given in Section 6.

Definition 2. A non-empty set $\boldsymbol{S}$ of atomic concepts is *consistent wrt. an LBN* $\langle n, \pi, \boldsymbol{L}, p \rangle$ if for all $i \in \{1, \ldots, n\}$, $|\boldsymbol{S} \cap \boldsymbol{L}_i| \leq 1$.[7] If $\boldsymbol{S}$ is consistent wrt. $\langle n, \pi, \boldsymbol{L}, p \rangle$ and $\boldsymbol{S} \subseteq \boldsymbol{L}$, let $(L_1, \ldots, L_m)$ enumerate the members of $\boldsymbol{S}$ in the order of the label sets of which they are members. Then $p(\sqcap \boldsymbol{S}) = p(L_1, \ldots, L_m)$.

5.2 SEMANTICS

The models for TDL presented in section 4.2 provide a basis for interpreting the meaning of labeled Bayesian networks, in as far as they represent the atomic expressions that occur in the label sets of such networks. The notion of a model needs to be extended, however, if it is to capture the probabilistic attributes of LBN's. The definition below encapsulates such an extension in the form of a *probability measure*, which for the purposes of this

[4] This could be arranged by using only a finite number of primitive concepts and roles, and placing an upper bound on the nesting of role quantification.

[5] Proof of this and other results may be found in (Yelland 1999).

[6] Note that while the sets of objects associated with a particular node must be disjoint, the sets associated with distinct nodes need not be.

[7] If $\boldsymbol{S}$ is not consistent wrt. the LBN, then by virtue of the fact that it contains two labels from the same node, the intersection of the extensions of the concepts it comprises is necessarily empty in any model of the network.

paper can be thought of simply as a function that assigns a probability to sets of objects—see (Halmos 1974) for details. Intuitively, the value assigned a set by the probability measure of such a model represents the probability that a randomly-picked object will be a member of that set.

Definition 3. A *probabilistic TDL (PTDL) model* $\langle D,c,r,[\_],\mu\rangle$ is a TDL model $\langle D,c,r,[\_]\rangle$, together with a probability measure μ mapping subsets of D to the interval [0,1], such that the denotation $[\![C]\!]$ of every concept expression C in TDL is in the domain of μ.

The relationship between a network and its models derives from the probabilities they ascribe to sets of atomic concepts. Specifically, for a probabilistic TDL model to satisfy a labeled Bayesian network, it must:

- Assign a probability to a conjunction of the atomic concepts in a network that agrees with the probability assigned to that conjunction by the network itself.
- Ensure that an object's membership of concepts in the network is *independent* of its membership of concepts outside of the network. This means that the probability that an object is a member of a conjunction with components inside and outside the network can be computed simply by multiplying the probability that it's a member of all the components inside the network by the probability of membership of those components outside.

Definition 4. A PTDL model $\langle D,c,r,[\_],\mu\rangle$ *satisfies* an LBN $\langle n,\pi,L,p\rangle$ (written $\langle D,c,r,[\_],\mu\rangle \models \langle n,\pi,L,p\rangle$) iff the following hold:

1. For all non-empty $S \subseteq L$ consistent wrt. $\langle n,\pi,L,p\rangle$, $\mu([\![\bigsqcap S]\!]) = p(\bigsqcap S)$
2. For all consistent S such that $S \cap L$ is non-empty, $\mu([\![\bigsqcap (S \setminus L)]\!]) \neq 0$ and $\mu([\![\bigsqcap S]\!]) = p(\bigsqcap (S \cap L)) \times \mu([\![\bigsqcap (S \setminus L)]\!])$

5.3 COHERENCE

Thus far, a framework has been proposed for labeling Bayesian networks with description logic expressions. It was pointed out in section 3, however, that it is all too easy to produce inconsistent specifications in this manner. While the definition of TDL and the restriction to atomic concepts diminish the opportunities for inconsistency, they are not eliminated. The following lays down a condition sufficient to rule out inconsistency entirely.

Definition 5. A labeled Bayesian network $\langle n,\pi,L,p\rangle$ is *coherent* iff $p(\bar{L}) = 0$ for all $\bar{L} \in L_1 \times \ldots \times L_n$ such that

1. $\bar{L} = \langle \ldots, (R_1 : \ldots : R_m : P), \ldots, (R_1 : \ldots : R_m : P'), \ldots \rangle$, for $P \neq P'$ and
2. $P, P' \in L_i$, for some $i \in \{1,\ldots,n\}$.

If an LBN meets the above requirement, inference can proceed from it ensured of well-groundedness. Formally:

Proposition 2. All coherent LBN's have at least one non-trivial model.

6 MAKING INFERENCES

Fundamental to the application of the framework described above is the assessment of conditional probabilities involving general (as distinct from atomic) concepts, that is, the probability that an object is a member of one concept, given that it is a member of another. This operation—termed *probabilistic subsumption* in (Koller, Levy and Pfeffer 1997)—is also a central feature of the P-CLASSIC system.

In the following, the probability that an arbitrary object is a member of (general) concept C_1 given that it is a member of (general concept) C_2 is denoted $\lambda(C_1 \mid C_2)$. With respect to a given PTDL model $\langle D,c,r,[\_],\mu\rangle$, $\lambda(C_1 \mid C_2)$ is defined as the ratio of the joint probability of C_1 and C_2 to the probability of C_2. In symbols:

$$\langle D,c,r,[\_],\mu\rangle \models \lambda(C_1 \mid C_2) = \lambda \quad \text{iff } \mu([\![C_1]\!] \cap [\![C_2]\!]) = \lambda \times \mu([\![C_2]\!])$$

There is a particular class of conditional probability assertion for which a simple inference procedure can be readily formulated. Assume we are given a coherent LBN $\mathcal{B} = \langle n,\pi,L,p\rangle$ and two TDL concepts C_1 and C_2. Further, (from Proposition 1) assume that $\bigsqcap S_1 = \nu(C_1)$ and $\bigsqcap S_2 = \nu(C_2)$, where $S_1 \subseteq L$, and both S_1 and S_2 are consistent wrt. $\mathcal{B}$. Define $S_2' = S_2 \cap L$. Then provided $p(\bigsqcap S_2') \neq 0$, we assert:

$$\langle n,\pi,L,p\rangle \vdash \lambda(C_1 \mid C_2) = \frac{p(\bigsqcap (S_1 \cup S_2'))}{p(\bigsqcap S_2')}$$

In computing the probabilities on the right hand side of the above equation, any of the algorithms for inference in Bayesian networks mentioned in section 2.1 may be employed.

It is straightforward to establish the soundness of the above procedure (Yelland 1999); a network entails a conditional probability according to the procedure only if that probability holds in all models of the network:

$$\langle n,\pi,L,p\rangle \vdash \lambda(C_1 \mid C_2) = \lambda \Rightarrow$$
$$\forall\langle D,c,r,[\_],\mu\rangle.\ \langle D,c,r,[\_],\mu\rangle \models \langle n,\pi,L,p\rangle \Rightarrow$$
$$\langle D,c,r,[\_],\mu\rangle \vdash \lambda(C_1 \mid C_2) = \lambda$$

7 APPLICATION

The primary impetus for the development of the representation framework arose from work carried out at Sun Microsystems forecasting demand for new product introductions. Initially, this involved the construction of Bayesian networks using judgmental information elicited from experts in the company. As the process was repeated for a number of products, it became apparent that certain elements of the networks produced were congruent, in as far as they related to common product and market features. Using the system described here, such common features and their ramifications may be captured as atomic concepts in an LBN, so that the inference procedure described above can be applied to compound concepts that comprise them.

By way of illustration, consider the (greatly simplified) example below. Here, the intent is to determine target market segments for server products offering a range of price/performance combinations.

The products themselves are described by concept expressions in TDL, composed from the primitive concepts and roles described informally in Figure 1. Figure 2 lists expressions that formally characterize the defined concepts in the previous figure.

In Figure 3, a labeled Bayesian network is displayed that relates certain properties of products to their ideal market segments. For example, the network suggests that an expensive product with high storage capacity is most likely a product for large businesses; conversely, an expensive product with low storage capacity is probably a dud.

Given the information above, the inference procedure in the previous section may be employed to estimate various aspects of the market for the new server products. For instance, the suitability of the Gold model for sale as an enterprise product is given by the expression:

$$\lambda(\text{EntPdct} \mid \text{Gold})$$

Primitive Concepts	
StorageUnit	— *The set of all storage products*
Server	— *The set of all server products*
High, Medium, Low	— *Quantity ranges*
LargeBusiness, SmallBusiness	— *Market segments*
Dud	— *A product that's not likely to sell anywhere*
Roles	
Capacity	— *Capacity of a storage unit*
Speed	— *Speed of a storage unit*
Tech	— *Quality of innovation in a server product*
Price	— *Price of a server product*
Storage	— *The type of storage unit in a server*
MktSegment	— *Ideal market segment for a server product*
Defined Concepts	
S10	— *Low-end storage unit*
S100	— *High-end storage unit*
Gemstone	— *A high-end server line (not necessarily expensive)*
Metal	— *A low-end server line (cheap)*
Opal	— *A server in the* Gemstone *line with an* S10 *storage unit*
Sapphire	— *An expensive server in the* Gemstone *line with an* S100 *storage unit*
Silver	— *A server in the* Metal *line with an* S10 *storage unit*
Gold	— *A server in the* Metal *line with an* S100 *storage unit*
EntPdct	— *A product best suited for sale into large businesses*
DeptPdct	— *A product best suited for sale into medium business*

Figure 1: Informal Descriptions

To compute the value of this expression, first express each concept in terms of its atomic constituents, as shown in Proposition 1:

$$\begin{aligned}\nu(\text{EntPdct}) &= \text{MktSegment:LargeBusiness}\\ \nu(\text{Gold}) &= \text{Server} \sqcap \text{Tech:Low} \sqcap \text{Price:Low} \sqcap\\ &\quad \text{Storage:StorageUnit} \sqcap\\ &\quad \text{Storage:Capacity:High} \sqcap\\ &\quad \text{Storage:Speed:High}\end{aligned}$$

This gives:

$$S_1 = \{\text{MktSegment:LargeBusiness}\}$$

$$S_2 = \left\{\begin{array}{l}\text{Server, Tech:Low, Price:Low,}\\ \text{Storage:StorageUnit,}\\ \text{Storage:Capacity:High,}\\ \text{Storage:Speed:High}\end{array}\right\}$$

$$S_2' = \{\text{Price:Low, Storage:Capacity:High}\}$$

S10 = StorageUnit ⊓ Capacity:Low ⊓ Speed:Medium
S100 = StorageUnit ⊓ Capacity:High ⊓ Speed:High
Gemstone = Server ⊓ Tech:High
Metal = Server ⊓ Tech:Low ⊓ Price:Low
Opal = Gemstone ⊓ Storage:S10
Sapphire = Gemstone ⊓ Storage:S100 ⊓ Price:High
Silver = Metal ⊓ Storage:S10
Gold = Metal ⊓ Storage:S100
EntPdct = MktSegment:LargeBusiness
DeptPdct = MktSegment:MediumBusiness

Figure 2: Formal Concept Definitions

Storage:Capacity:High	0.4
Storage:Capacity:Low	0.6

Price:High	0.6
Price:Low	0.4

	Mkt-Segment: Large-Business	Mkt-Segment: Small-Business	Dud
Storage:Capacity:High Price:High	0.7	0.1	0.2
Storage:Capacity:High Price:Low	0.8	0.2	0
Storage:Capacity:Low Price:High	0.2	0.1	0.7
Storage:Capacity:Low Price:Low	0.2	0.4	0.4

Figure 3: Labeled Bayesian Network

It is not difficult to verify that all the preconditions set out in section 6 are satisfied in this case. Hence by the inference rule:

$$\lambda(\text{EntPdct} \mid \text{Gold}) = \frac{p\left(\bigsqcap \left\{\begin{array}{l}\text{MktSegment:LargeBusiness},\\ \text{Price:Low},\\ \text{Storage:Capacity:High}\end{array}\right\}\right)}{p\left(\bigsqcap\{\text{Price:Low}, \text{Storage:Capacity:High}\}\right)} = \frac{0.8 \times 0.4 \times 0.4}{0.4 \times 0.4} = 0.8$$

Of course, this example is somewhat unrealistic in its simplicity. In practice, the results of the inference rule are used to estimate parameters for market models such as that described in (Bass 1969) and (Lilien and Rangaswamy 1997). Model averaging (Singpurwalla and Wilson 1999) is used to weight the predictions derived from the models according to the probabilities associated with particular parameter values. For a more complete (though informal) description of this approach to demand forecasting, see (Thomas 1993).

8 RELATED WORK

Essays in the combination of formal logic and probability have a long history: Some of the later research of Carnap (1962), for example, centers on the topic. More recently, Bacchus (1990) and Halpern (1990) have sought to fashion modal logics encompassing probabilistic reasoning. Such general frameworks are useful in elucidating judgments regarding probabilities, but they give rise to inference procedures too complex to mechanize effectively (Abadi and Halpern 1994).

A number of researchers have turned to description logics as more tractable basis for probabilistic inference; in this connection, the work most closely allied to that described here is P-CLASSIC.

8.1 P-CLASSIC

Introduced in (Koller, Levy and Pfeffer 1997), the intent of P-CLASSIC is to augment an existing description logic—the CLASSIC system described in (Brachman et al. 1991)—with the facilities of Bayesian networks. The central contribution of P-Classic is the *probabilistic class*, or *p-class*. A p-class resembles a labeled Bayesian network (as defined in section 5), in that it consists of a Bayesian network whose nodes are associated with concept expressions. The intent is to assert a degree of correlation (or "overlap," as it is termed in P-CLASSIC) between certain concepts by specifying a joint probability distribution involving those concepts.

Primitive concepts may label the nodes of a p-class, as with LBN's. In contrast to LBN's, however (where they may appear on a node in the company of arbitrary atomic expressions), a primitive concept may only appear in a p-class on a node whose only other label is the negation of that concept.[8]

The most profound difference between the approach taken in this paper and that of P-CLASSIC occurs in the treatment of role quantification. Note that in an LBN, it is possible to specify explicitly dependencies between arbitrary atomic concept expressions, including concepts involving role quantification, such as $R:P$.

P-CLASSIC handles such dependencies much more indirectly: In P-CLASSIC, a p-class ρ contains for each role R,

[8] P-CLASSIC extends CLASSIC with negations of primitive concepts.

a probabilistic node NUMBER(R), and a deterministic node $PC(R)$.

For each configuration φ of nodes in the network labeled by primitive concepts, and each number n less than b_R (an arbitrary finite upper bound that P-CLASSIC requires is imposed on the number of R-fillers of any object), node NUMBER(R) specifies the probability that an object of p-class ρ will have n R-fillers, which probability is denoted $\Pr_\rho(\text{NUMBER}(R) = n \mid \varphi)$.

The deterministic node $PC(R)$ uniquely selects some other p-class, $\rho(R,n,\varphi)$, for each number n of R-fillers and configuration φ. P-class $\rho(R,n,\varphi)$ in turn defines a probability $\Pr_{\rho(R,n,\varphi)}(C)$ for concept C.

The occupancy of R-fillers is assumed independent in P-CLASSIC, so that according to the original p-class ρ, the probability (given configuration φ) that an object has n R-fillers in concept C is:

$$\Pr_\rho(\text{NUMBER}(R) = n \mid \varphi) \times \left[\Pr_{\rho(R,n,\varphi)}(C)\right]^n$$

Therefore, the probability assigned the concept $\forall R : C$ is calculated as:

$$\sum_{n=1}^{b_R} \left(\Pr_\rho(\text{NUMBER}(R) = n \mid \varphi) \times \left[\Pr_{\rho(R,n,\varphi)}(C)\right]^n \right).$$

This approach has intuitive appeal, but it makes it very difficult to express relationships between concepts involving role quantification—particularly since (at least as described in (Koller, Levy and Pfeffer 1997)) P-CLASSIC insists that NUMBER(R) may be the parent only of $PC(R)$, and that $PC(R)$ be the parent of no other node. In fact, the semantics of P-CLASSIC require that the properties of an object and its fillers are independent given the filler's p-class, and so it is impossible to express the sort of assertions that figure in the sample application given above. By contrast, the framework described in this paper shares none of these restrictions, though (as was stated above) in part, this generality results from its use of a description logic component significantly less capable than P-CLASSIC's.

8.2 HEINSOHN

In (Heinsohn 1991), Heinsohn highlights a probabilistic extension to description logics developed in full in his Ph.D. thesis (Heinsohn 1993). Heinsohn's system extends a simple description logic (a minor generalization of the language $\mathcal{FL}^-$ defined in section 2.2) with *p-conditionings*. A p-conditioning involving concepts C_1 and C_2 is an assertion the form:

$$C_1 \xrightarrow{[p_l, p_u]} C_2, \quad \text{where } 0 \le p_l \le p_u \le 1.$$

Such a p-conditioning imposes a constraint on the cardinalities of the extensions of C_1 and C_2 such that:

$$\frac{\left|[\![C_1]\!] \cap [\![C_2]\!]\right|}{\left|[\![C_1]\!]\right|} \in [p_l, p_u]$$

(Heinsohn's framework requires that the domain of all objects is finite.) Heinsohn provides inference rules that allow new p-conditionings to be derived in the presence of concept definitions.

8.3 JAEGER

Jaeger's work, summarized in (Jaeger 1994), is closely allied with Heinsohn's, though Jaeger places rather greater emphasis on the derivation of beliefs about individual objects in the presence of statistical assertions. (The work detailed in this paper focuses exclusively on statistical assertions; for an introduction to the treatment of individuals, see (Halpern 1990).) In Jaeger's framework, general probabilistic assertions of the form $p(C \mid D)$, for (possibly non-atomic) concept expressions C and D, impose linear equalities on probability measures defined over the *Lindenbaum algebra* of concept expressions.[9] Jaeger shows how such equalities determine a convex polytope (in the space of discrete probability distributions), which may be used to derive bounds on the conditional probabilities of other concept expressions.

Both Heinsohn and Jaeger's approach to probabilistic inference (viz., the derivation of probability intervals from partial descriptions of conditional probabilities) resemble the rule-based approaches of some early expert systems (Neapolitan 1990). This sort of framework is very general, and makes it easy to establish the basics of a knowledge base, but it suffers in certain regards in comparison with the Bayesian network approach espoused here. For example, as is shown in (Castillo, Gutiérrez and Hadi 1997) it can be very difficult to establish *a priori* that a set of conditional probability assertions of this nature is actually consistent; in contrast, any labeled Bayesian network that satisfies the fairly simple constraints in Definition 5 is consistent by construction. Furthermore, inferences based on Bayesian networks can often be carried out much more efficiently than in the more general Heinsohn/Jaeger framework, as Koller, Levy and Pfeffer (1997) point out.

[9] The Lindenbaum algebra is formed by taking the quotient of the set of all concept expressions modulo logical equivalence.

9 CONCLUSION

This paper has outlined an approach to the combination of Bayesian networks and description logics that aims to fortify the capabilities of Bayesian networks using a simple description logic. This contrasts with the most closely-related work—viz. P-CLASSIC—that begins with an existing description logic, extending it with the facilities offered by Bayesian networks. By trading away a degree of expressiveness in its description logic component, the system described here is able to circumvent many of the restrictions that apply to applications using P-CLASSIC.

Thought it has proven quite adequate in practical applications, it is incontestable that the description logic TDL provides only the barest essentials required of such a notation. It would be interesting to explore the possibility of incorporating features found in more expressive logics, such as the description logic CLASSIC, for example, that underlies P-CLASSIC. Whether the introduction of such features (absent the restrictions of P-CLASSIC) is compatible with the fully determinate probabilities and simple consistency conditions offered by the current configuration is an open question, however. It may be that a more complex, interval-based approach to probabilities like those of Heinsohn and Jaeger is required.

10 REFERENCES

Abadi, M. and Halpern, J. (1994) Decidability and expressiveness for first-order logics of probability. *Information and Computation* 112:1, pp. 1-36.

Bacchus, F. (1990) *Representing and Reasoning with Probabilistic Knowledge*, MIT Press, Cambridge, MA.

Bass, F. M. (1969) A new product growth model for consumer durables, *Management Science*, Vol. 15, No. 4 (January), pp. 216-227.

Brachman, R. J. and Levesque, H. J. (1984) The tractability of subsumption in frame-based description languages. *Proceedings of the 4th National Conference on Artificial Intelligence (AAAI-84).*

Brachman, R. J. and Schmolze, J. G. (1985) An overview of the KL-ONE knowledge representation system. *Cognitive Science*, Vol. 9, No. 2.

Brachman, R., Borgida, A., McGuinness, D., Patel-Schneider, P. and Resnick, L., (1991) Living with CLASSIC: When and how to use a KL-ONE-like language. In Sowa, J., (*ed.*), *Principles of Semantic Networks*, Morgan Kaufmann.

Carnap, R. (1962) *Logical Foundations of Probability*, University of Chicago Press, Chicago, IL, 2 edition.

Castillo, E., Gutiérrez, J. and Hadi, A. (1997) *Expert Systems and Probabilistic Network Models.* Monographs in Computer Science, Springer-Verlag, New York, NY.

Halmos, P. (1974) *Measure Theory*. Graduate Texts in Mathematics, 18, Springer-Verlag, New York, NY.

Halpern, J. (1990) An analysis of first-order logics of probability, *Artificial Intelligence* 46, pp. 311-350.

Heinsohn, J. (1991) *A Hybrid Approach for Modeling Uncertainty in Terminological Logics*, Deutsche Forschungszentrum für Künstliche Intelligenz Research Report-1991-24.

Heinsohn, J. (1993) *ALCP—Ein hybrider Ansatz zur Modellierung von Unsicherheit in terminologischen Logiken.* Ph.D. thesis, Universität des Saarlandes, Saarbrücken.

Jaeger, M. (1994) A logic for default reasoning about probabilities, in *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, R. Lopez de Mantaras and D. Poole (editors), Morgan Kaufmann Publishers, San Francisco, CA.

Jensen, F. V. (1996) *An Introduction to Bayesian Networks*. Springer, New York.

Koller, D., Levy, A. and Pfeffer, A. (1997) P-CLASSIC: A tractable probabilistic description logic. *Proceedings of 14th National Conference on Artificial Intelligence (AAAI-97)*, pp. 360-397.

Lilien, G. L. and Rangaswamy, A. (1997) *Marketing Engineering: Computer-Assisted Marketing Analysis and Planning*, Addison-Wesley, Reading, MA.

MacGregor, R (1991) The evolving technology of classification-based knowledge representation systems. Sowa, J. F., (ed.) *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Morgan Kaufmann, San Mateo.

Mantay, T. and Möller, R. (1998) Content-based information retrieval by computation of least common subsumers in a probabilistic description logic. *Proc. International Workshop on Intelligent Information Integration, ECAI'98*, Aug. 23-28, Brighton, UK.

Moss, L. S. (1991) Completeness theorems for logics of feature structures. Y.N. Moschovakis (ed.), *Logic in Computer Science*, MSRI Publications Vol. 21, Springer-Verlag, pp. 387-401.

Neapolitan, R. E. (1990) *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*, Wiley-Interscience, New York, NY.

Pierce B. C. (1991) *Programming with Intersection Types and Bounded Polymorphism*. Technical Report CMU-CS-91-205, Carnegie-Mellon University.

Singpurwalla, N. D. and Wilson, S. P. (1999) *Statistical Methods in Software Engineering: Reliability and Risk*, Springer-Verlag, New York, NY.

Thomas, R. J. (1993) *New Product Development: Managing and Forecasting for Strategic Success*, John Wiley & Sons, New York, NY.

Yelland, P. M. (1999) *Market Analysis Using a Combination of Bayesian Networks and Description Logics*, Sun Microsystems Laboratories Tech. Rep. TR-99-0241.

Zhang, N. L. and Poole, D. (1996) Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, 5.

Independence in Qualitative Uncertainty Frameworks

N. Ben Amor
ISG Tunis*
nahla.benamor@ihec.rnu.tn

S. Benferhat
I.R.I.T†
benferhat@irit.fr

D. Dubois
I.R.I.T
dubois@irit.fr

H. Geffner
USB‡
hector@usb.ve

H. Prade
I.R.I.T
prade@irit.fr

Abstract

The notion of independence is central in many information processing areas, such as multiple criteria decision making, databases organization, or uncertain reasoning. This is especially true in the later case where the success of Bayesian networks is basically due to the graphical representation of independence they provide. This paper first studies qualitative independence relations when uncertainty is encoded by a complete pre-order between states of the world. While a lot of work has focused on the formulation of suitable definitions of independence in different qualitative uncertainty frameworks, our interest in this paper is rather to formulate a general definition of independence based on pure ordering considerations, and that applies to all qualitative uncertainty frameworks. The second part of the paper investigates the impact of the embedding of qualitative independence relations into qualitative uncertainty frameworks such as possibility theory, or Spohn functions. The absolute scale used for grading uncertainty in these settings enforces the commensurateness between local pre-orders (since they share the same scale). This leads to an easy decomposability property of the joint distributions into more elementary relations on the basis of the independence relations.

*Institut Supérieur de Gestion Tunis, Tunisia.
†Institut de Recherche en Informatique de Toulouse, 118, Route de Narbonne, 31062 Toulouse Cedex 04. France.
‡Universidad Simon Bolivar, Departamento de Computacion, Caracas, Venezuela.

1 Introduction

Independence relations between variables play an important role in the handling of uncertain information. Indeed various notions of (in)dependence are central in multiple criteria analysis, in data decomposition, or in uncertain reasoning based on Bayesian networks or logical reasoning. There has been a considerable interest in artificial intelligence in the last few years for discussing independence in various representation frameworks.

Bayesian independence and database functional dependencies have been formally related (Wong et al. 1995). Independence in the logical setting has received a lot of attention, e.g., (Darwiche 1997), (Lakemeyer 1997), (Lang and Marquis 1998). In this paper we rather investigate possible definitions of independence in qualitative settings, using qualitative uncertainty relations, or their graded counterparts such as possibility measures for instance.

From an operational point of view, two forms of independence can be distinguished:

- *decomposition independence* which allows the decomposition of a joint distribution pertaining to tuples of variables into local distributions on smaller subsets of variables, in order to have a reasoning machinery working at a local level without losing any information.

- *causal independence* for expressing the absence of causality. This form of independence is always characterized in semantic terms, e.g. two variables (or sets of variables) are said to be independent if our belief in the value of one of them does not change when learning something about the value of the other.

These two kinds of independence are not necessarily mutually exclusive. Ideally, a good definition of independence both expresses the lack of causality and is useful for computations.

In Section 2, we first present a general qualitative uncertainty framework where uncertainty is represented by total pre-orders on different subsets of situations. Then, in Section 3, we study several definitions of qualitative independence (causal and decomposition ones). A noticeable causal qualitative independence is proposed. This definition extends the one proposed by Darwiche (Darwiche 1997) to qualitative uncertainty frameworks. Finally, we discuss the advantages of representing a total pre-order with ranking functions. In Section 5 after providing a background on ranking functions frameworks, we show that this is crucial for decomposing joint distributions on the basis of independence relations. Indeed, some qualitative uncertainty relations ordered with respect to well-known principles (like leximin or leximax ordering) cannot be decomposable in the qualitative setting while they become decomposable with the help of ranking functions which make these relations commensurable. Lastly, in Section 6, we provide a comparative study between existing possibilistic independence and the qualitative independence relations proposed in this paper.

2 Qualitative uncertainty framework

2.1 Notations

Let $U = \{A, B, C, ...\}$ be the set of variables. We denote by $D_A = \{a_1, ..., a_n\}$ the supposedly finite domain associated to the variable A. By a we denote any instance of A. $X, Y, Z, ...$ denote disjoint subsets of variables in U, and $D_X = \{x_1, x_2, ..., x_m\}$ represents the Cartesian product of variable domains in X. By x we denote any instance of X. Ω denotes the universe of discourse, which is the Cartesian product of all variable domains in U. Each element of Ω is called interpretation, situation or state of the world. ϕ, ψ denote the subsets of Ω, called formula (or event) and $\neg\phi$ denotes the complementary set of ϕ i.e. $\neg\phi = \Omega - \phi$.

2.2 Representation of uncertain information

In the following, we give a formal description of the qualitative representation of uncertainty we are using. The basic idea is to represent the available incomplete information of the real world by a **total pre-order** [1], on Ω. This total pre-order called a *qualitative plausibility relation*, will be denoted by $\geq_\Omega$. The relation $\omega_1 \geq_\Omega \omega_2$ means that ω_1 is more plausible than ω_2. We denote $=_\Omega$ and $>_\Omega$ respectively the equality and the strict inequality relations associated with $\geq_\Omega$.

Given a relation $\geq_\Omega$ on Ω, we can lift it to another plausibility relation defined on the subsets of Ω (for the sake of simplicity, we use the same notation $\geq_\Omega$) by (e.g., (Dubois 1986)): $\phi \geq_\Omega \psi$ iff $\forall \omega \in \psi, \exists \omega' \in \phi$ such that $\omega' \geq_\Omega \omega$. Namely, $\phi \geq_\Omega \psi$ holds if the best element in ϕ is preferred to the best element(s) in ψ.

This qualitative representation of uncertainty is used in several non-monotonic formalisms like Lehmann's ranked model (Lehmann 1989), plausibility relations (Halpern 1997), Spohn's ordinal conditional functions (Spohn 1988) and possibility theory (Dubois and Prade 1988, 1998).

2.3 Qualitative conditioning

Conditioning is a crucial notion in studying independence relations. In the qualitative setting, it consists in transforming a plausibility relation $\geq_\Omega$ on the basis of a new information $\phi \subseteq \Omega$ into a new plausibility relation denoted by $\geq_{\Omega|\phi}$. This new relation is obtained by applying the three following postulates

$\mathbf{A_1}$: $\forall \omega_1, \omega_2 \in \phi, \omega_1 >_\Omega \omega_2$ iff $\omega_1 >_{\Omega|\phi} \omega_2$,

$\mathbf{A_2}$: $\forall \omega_1 \in \phi, \forall \omega_2 \notin \phi, \omega_1 >_{\Omega|\phi} \omega_2$,

$\mathbf{A_3}$: $\forall \omega_1, \omega_2 \notin \phi, \omega_1 =_{\Omega|\phi} \omega_2$.

$\mathbf{A_1}$ means that the new plausibility relation should not alter the initial order between the elements of ϕ. $\mathbf{A_2}$ confirms that each interpretation of ϕ should be preferred to any interpretation not belonging to ϕ. Finally, the last postulate $\mathbf{A_3}$ says that the elements not belonging to ϕ are in the same equivalence class. These three postulates determine the new plausibility relation $\geq_{\Omega|\phi}$ in a **unique** manner.

2.4 Accepted beliefs

We now introduce a further definition which will be helpful in easily defining the notion of qualitative independence, namely the notion of accepted belief, e.g., (Dubois and Prade 1995), (Dubois et al., 1996), (Dubois et al., 1997a), (Friedman and Halpern, 1995, 1996).

Definition 1 *An acceptance function associated to a plausibility relation $\geq_\Omega$ is a function, denoted by*

[1] a relation $\geq_\Omega$ on Ω is a total pre-order if $\geq_\Omega$ is reflexive, transitive and complete, i.e., for all ω_1, ω_2, we have: $\omega_1 \geq_\Omega \omega_2$ or $\omega_2 \geq_\Omega \omega_1$.

$\mathbf{Acc}_{\geq_\Omega}(.)$, which assigns to each ϕ a value in $\{-1,0,1\}$ in the following way:

$$\mathbf{Acc}_{\geq_\Omega}(\phi) = \begin{cases} 1 & \text{if } \phi >_\Omega \neg\phi \\ 0 & \text{if } \phi =_\Omega \neg\phi \\ -1 & \text{if } \neg\phi >_\Omega \phi. \end{cases}$$

When $\mathbf{Acc}_{\geq_\Omega}(\phi) = 1$ (resp. $\mathbf{Acc}_{\geq_\Omega}(\phi) = -1$) we say that ϕ is accepted (resp. rejected). $\mathbf{Acc}_{\geq_\Omega}(\phi) = \mathbf{Acc}_{\geq_\Omega}(\neg\phi) = 0$, corresponds to the situation of total ignorance concerning ϕ, i.e., ϕ and $\neg\phi$ are equally plausible.

The function $\mathbf{Acc}_{\geq_\Omega}$ can be extended in order to take into account a given context. Then a conditional belief measure denoted by $\mathbf{Acc}_{\geq_\Omega}(.|.)$ is defined by

$$\mathbf{Acc}_{\geq_\Omega}(\phi \mid \psi) = \begin{cases} 1 & \text{if } \phi \cap \psi >_\Omega \neg\phi \cap \psi \\ 0 & \text{if } \phi \cap \psi =_\Omega \neg\phi \cap \psi \\ -1 & \text{if } \neg\phi \cap \psi >_\Omega \phi \cap \psi. \end{cases}$$

Remarks:

- The plausibility relation $\geq_\Omega$ determines $\mathbf{Acc}_{\geq_\Omega}$ in a unique manner. The converse is not true. Namely, many plausibility relations can generate a same set of **plain beliefs**, i.e, can have the same $\mathbf{Acc}_{\geq_\Omega}$ on all formulas(including the interpretations).

  Counter-example : Let us consider the following values of $\mathbf{Acc}_{\geq_\Omega}$ relative to the two binary variables A and B:
  $\mathbf{Acc}_{\geq_\Omega}(a_1) = \mathbf{Acc}_{\geq_\Omega}(b_1) = 1$,
  $\mathbf{Acc}_{\geq_\Omega}(a_2) = \mathbf{Acc}_{\geq_\Omega}(b_2) = -1$,
  $\mathbf{Acc}_{\geq_\Omega}(a_1 \vee b_1) = 1, \mathbf{Acc}_{\geq_\Omega}(a_2 \vee b_1) = 1$,
  $\mathbf{Acc}_{\geq_\Omega}(a_1 \vee b_2) = 1, \mathbf{Acc}_{\geq_\Omega}(a_2 \vee b_2) = -1$,
  $\mathbf{Acc}_{\geq_\Omega}(a_2 \wedge b_1) = -1, \mathbf{Acc}_{\geq_\Omega}(a_2 \wedge b_2) = -1$,
  $\mathbf{Acc}_{\geq_\Omega}(a_1 \wedge b_2) = -1, \mathbf{Acc}_{\geq_\Omega}(a_1 \wedge b_1) = 1$.

  We can check that the two following plausibility relations:
  $a_1 \wedge b_1 >_{\Omega_1} a_2 \wedge b_1 >_{\Omega_1} a_1 \wedge b_2 =_{\Omega_1} a_2 \wedge b_2$, and
  $a_1 \wedge b_1 >_{\Omega_2} a_2 \wedge b_1 >_{\Omega_2} a_1 \wedge b_2 >_{\Omega_2} a_2 \wedge b_2$
  generate the same information on the accepted beliefs as those given above i.e:

  $$\mathbf{Acc}_{\geq_{\Omega_1}} = \mathbf{Acc}_{\geq_{\Omega_2}} = \mathbf{Acc}_{\geq_\Omega}.$$

- The set of all **conditional beliefs** determines in a unique manner a plausibility relation on Ω constructed in this way:

  $$\omega_1 >_\Omega \omega_2 \text{ iff } \mathbf{Acc}_{\geq_\Omega}(\{\omega_1\} \mid \{\omega_1, \omega_2\}) = 1.$$

- The measures $\mathbf{Acc}_{\geq_\Omega}(.)$ and $\mathbf{Acc}_{\geq_\Omega}(.|.)$ are linked by the following equation

  $$\mathbf{Acc}_{\geq_\Omega}(\phi \wedge \psi) = min(\mathbf{Acc}_{\geq_\Omega}(\phi \mid \psi), \mathbf{Acc}_{\geq_\Omega}(\psi))$$

  which is similar to Bayes conditioning.

- $\mathbf{Acc}_{\geq_\Omega}(. \mid \phi)$ can be defined from $\mathbf{Acc}_{\geq_{\Omega|\phi}}$ in the following way:

  $$\forall\psi, \mathbf{Acc}_{\geq_\Omega}(\psi \mid \phi) = \mathbf{Acc}_{\geq_{\Omega|\phi}}(\psi).$$

In the following, we use $\mathbf{Acc}(.)$ (resp. $\mathbf{Acc}(.|\ .)$) instead of $\mathbf{Acc}_{\geq_\Omega}(.)$ (resp. $\mathbf{Acc}_{\geq_\Omega}(.|\ .)$) when there is no ambiguity.

3 Qualitative independence

3.1 In search of causal qualitative independence

Independence can be thought of either in terms of qualitative plausibility relations or in terms of acceptance measures. The two views can be related, as shown in this section where we present three (distinct) possible definitions of causal independence. Basically, two sets of variables X and Y are declared to be independent (in a way or another) if learning any instance of Y:

- preserves the preferred (or top) instances of X, or
- preserves the accepted (resp. rejected and ignored) instances of X, or
- preserves the relative ordering between instances of X.

3.1.1 Preserving preferred instances

The first idea is to consider a variable set X as independent of Y in the context Z if for all instance z of Z, the acceptance of any instance $(x \wedge y)$ of (X, Y) is fully determined by the separate acceptance of x and y. In particular, if x and y are accepted, then $(x \wedge y)$ is accepted. One way to relate the acceptance of $(x \wedge y)$ to the acceptance of x and the acceptance of y is:

$$\mathbf{Acc}(x \wedge y \mid z) = min(\mathbf{Acc}(x \mid z), \mathbf{Acc}(y \mid z)), \forall xyz. \quad (1)$$

It can be checked that this definition only cares about the preservation of the *top elements* (i.e. best elements) in X and Y in any context z of Z. In other terms, for any plausible instance x of X (i.e., $\mathrm{Acc}(x) = 1$) and for any plausible instance y of Y, $(x \wedge y)$

should be a plausible element of (X, Y). In the following, independence relations satisfying the equation (1) are called **PT**-independence (PT for Preserving Top elements), and are denoted by I_{PT}(X,Z,Y).

This is clearly a very weak definition of independence where generally the acceptance of one instance of X or of Y is enough to conclude the independence between these two variable sets. In particular, if a plausibility relation $\geq_\Omega$ contains exactly one preferred element then all variables are pairwise PT-independent.

3.1.2 Preserving accepted beliefs

One way of making the above definition stronger is to consider conditional beliefs. Namely, a variable set X is considered as independent of Y in the context Z, if for all instance z of Z the following condition is respected: any instance x of X is accepted (resp. rejected, ignored) in the context z and remains accepted (resp. rejected, ignored) in this context after knowing any instance y of Y. In other words, believing in X does not change when Y is learned. More formally X and Y are said to be **PB**-independent (PB for Preserving Beliefs) in the context Z, denoted by I_{PB}(X,Z,Y), if:

$$(i)\ \mathbf{Acc}(x \mid y \wedge z) = \mathbf{Acc}(x \mid z) \text{ and}$$
$$(ii)\ \mathbf{Acc}(y \mid x \wedge z) = \mathbf{Acc}(y \mid z), \forall xyz. \quad (2)$$

Note that contrarly to the situation in the probability theory, (i) and (ii) are not equivalent.

Counter-example : Consider two binary variables A and B with the following plausibility relation:

$$a_1 \wedge b_1 >_\Omega a_1 \wedge b_2 >_\Omega a_2 \wedge b_2 >_\Omega a_2 \wedge b_1$$

The relation (i) is true since $\mathbf{Acc}(a \mid b) = \mathbf{Acc}(a), \forall ab$, but (ii) is false since $\mathbf{Acc}(b_1) = 1 \neq \mathbf{Acc}(b_1 \mid a_2) = -1$.

In the case when the plausibility relation only contains two equivalent classes (which can, for instance, correspond to the models and counter-models of some classical databases), then this definition is equivalent to the so-called *logical conditional independence* (LCI) proposed by Darwiche (Darwiche 1997).

This definition of independence preserves the *acceptance* of instances of X in the context of Y but does not preserve the relative *ordering* between instances of X (except when we consider binary variables).

For instance let A and B be two independent variables (according to the previous definition) such that $D_A = \{a_1, a_2, a_3\}$ and $D_B = \{b_1, b_2\}$ with the following plausibility relation: $a_1 \wedge b_1 >_\Omega a_2 \wedge b_1 >_\Omega a_3 \wedge b_1 >_\Omega a_1 \wedge b_2 >_\Omega a_2 \wedge b_2 =_\Omega a_3 \wedge b_2$.

It can be checked that the relative ordering between instances of A is not preserved in context of B. Indeed, $a_1 >_\Omega a_2 >_\Omega a_3$ but $a_1 >_\Omega a_2 =_\Omega a_3$ in the context b_2.

3.1.3 Preserving the relative ordering

The definition, we propose now, simply says that two variable sets X and Y are independent in the context of Z, if for all z, the preferential ordering between the different values of X (resp. Y) is preserved after the revision by any value y of Y (resp. x of X). More formally:

Definition 2 *Let X, Y and Z be three disjoint subsets of U. X and Y are said to be PO-independent (PO for Preserving Ordering) in the context Z, denoted $I_{PO}(X,Z,Y)$, if $\forall z \in D_Z, y \in D_Y, x \in D_X$:*
(i) $\forall x_i, x_j \in D_X, x_i >_{\Omega|z} x_j$ iff $x_i >_{\Omega|y\wedge z} x_j$, and

$$(ii)\ \forall y_k, y_l \in D_Y, y_k >_{\Omega|z} y_l \text{ iff } y_k >_{\Omega|x\wedge z} y_l. \quad (3)$$

The following proposition rewrites PO-independence relations in terms of **Acc**.

Proposition 1 *X and Y are PO-independent in the context Z iff $\forall D_{X'} \subseteq D_X, \forall D_{Y'} \subseteq D_Y$ such that $D_{X'} \neq \emptyset$ and $D_{Y'} \neq \emptyset$ and $\forall xyz$:*

$$\boldsymbol{Acc}(x \wedge y \mid z, D_{X'}, D_{Y'}) =$$
$$min(\boldsymbol{Acc}(x \mid z, D_{X'}), \boldsymbol{Acc}(y \mid z, D_{Y'})). \quad (4)$$

As a corollary of this rewriting, we deduce:

Proposition 2 *If X and Y are PO-independent in the context Z, then they are PT-independent and PB-independent.*

Indeed equation (4) implies (1), since it is enough to let $D_{X'} = \Omega$ and $D_{Y'} = \Omega$ in (4) to obtain (1). It also implies (2) by letting $D_{X'} = \Omega$ and $D_{Y'} = \{y\}$ which leads to (i) of (2) and $D_{X'} = \{x\}$ and $D_{Y'} = \Omega$ which leads to (ii) of (2). Clearly, the converse of the above proposition, in general, does not hold. However, when we only restrict to binary variables then PO-independence and PB-independence are equivalent.

Lastly, we can check that PO-independence is equivalent to the independence relation based on *Ceteris Paribus* (all else being equal) principle used in (Boutilier et al. 1999) (Doyle and Wellman 1991). This principle is closely related to preferential independence in multiple criteria analysis, see, e.g.,(Bacchus and Grove 1996).

3.2 Decomposition independence based on remarkable plausibility relations

A natural way of defining decomposition independence relations is to analyze the structure of the plausibility relation $\geq_\Omega$. A plausibility relation is said to be decomposable w.r.t. X and Y in the context Z, iff $\geq_\Omega$ is a function of the local orderings $\geq_{X\cup Z}$ and $\geq_{Y\cup Z}$.

The following introduces a well known principle, called *Pareto-principle*, which defines a partial oredering between pairs $(x_i \wedge z, y_j \wedge z)$:

Definition 3 *Let X, Y, Z three disjoint sets of U. Then the pair $(x_i \wedge z, y_k \wedge z)$ is said to be Pareto-preferred to $(x_j \wedge z, y_l \wedge z)$, denoted by $(x_i \wedge z, y_k \wedge z) \geq_P (x_j \wedge z, y_l \wedge z)$, if and only if $x_i \wedge z \geq_\Omega x_j \wedge z$ and $y_k \wedge z \geq_\Omega y_l \wedge z$.*

In general $\geq_P$ is only a **partial** order. Since this paper deals with plausibility relations which are total pre-order, the following introduces a general class of plausibility relations which are compatible with the Pareto-principle:

Definition 4 *Let X, Y and Z be disjoint subsets of U. A plausibility relation $\geq_\Omega$ is said to be Pareto-compatible (or **monotonic**) on X and Y in the context Z if $\forall z \in D_Z, \forall x_i, x_j \in D_X, \forall y_k, y_l \in D_Y$, we have: $(x_i \wedge z, y_k \wedge z) \geq_P (x_j \wedge z, y_l \wedge z)$ implies $(x_i \wedge y_k \wedge z) \geq_\Omega (x_j \wedge y_l \wedge z)$.*

Well known orderings $>_\Omega$ used in the qualitative setting, which are Pareto-compatible, are Pareto-ordering, *leximin* and *leximax* orderings that we briefly present now (Moulin, 1988).

1. A plausibility relation $\geq_\Omega$ is said to be **Pareto-decomposable** on X and Y in the context Z, if $\forall z \in D_Z, \forall x_i, x_j \in D_X, \forall y_k, y_l \in D_Y$, we have:
   $x_i \wedge y_k \wedge z \geq_\Omega x_j \wedge y_l \wedge z$ iff
   $x_i \wedge z \geq_\Omega x_j \wedge z$ and $y_k \wedge z \geq_\Omega y_l \wedge z$.

   As we will see later, this definition is very strong since it implies that one of the local distributions on X or Y should be uniform. One can weaken this definition in the following way: $\geq_\Omega$ is said to be **strict Pareto-decomposable** on X and Y in the context Z, if $\forall z \in D_Z, \forall x_i, x_j \in D_X, \forall y_k, y_l \in D_Y$, we have:
   $x_i \wedge y_k \wedge z >_\Omega x_j \wedge y_l \wedge z$ iff
   (i) $x_i \wedge z \geq_\Omega x_j \wedge z$ and $y_k \wedge z \geq_\Omega y_l \wedge z$, and
   (ii) $x_i \wedge z >_\Omega x_j \wedge z$ or $y_k \wedge z >_\Omega y_l \wedge z$.

   An example of plausibility relation which is strict Pareto-decomposable but not Pareto-decomposable is the following one defined on two binary variables A and B:
   $$a_1b_1 >_\Omega a_1b_2 = a_2b_1 >_\Omega a_2b_2.$$

2. A plausibility relation $\geq_\Omega$ is said to be **leximin-decomposable** on X and Y in the context Z, if $\forall z \in D_Z, \forall x_i, x_j \in D_X, \forall y_k, y_l \in D_Y$, we have:
   $x_i \wedge y_k \wedge z >_\Omega x_j \wedge y_l \wedge z$ iff
   (i) $min(x_i \wedge z, y_k \wedge z) >_\Omega min(x_j \wedge z, y_l \wedge z)$ or
   (ii) $min(x_i \wedge z, y_k \wedge z) =_\Omega min(x_j \wedge z, y_l \wedge z)$ and $max(x_i \wedge z, y_k \wedge z) >_\Omega max(x_j \wedge z, y_l \wedge z)$.[2]

3. A plausibility relation $\geq_\Omega$ is said to be **leximax-decomposable** on X and Y in the context Z, if $\forall z \in D_Z, \forall x_i, x_j \in D_X, \forall y_k, y_l \in D_Y$, we have:
   $x_i \wedge y_k \wedge z >_\Omega x_j \wedge y_l \wedge z$ iff
   (i) $max(x_i \wedge z, y_k \wedge z) >_\Omega max(x_j \wedge z, y_l \wedge z)$ or
   (ii) $max(x_i \wedge z, y_k \wedge z) =_\Omega max(x_j \wedge z, y_l \wedge z)$ and $min(x_i \wedge z, y_k \wedge z) >_\Omega min(x_j \wedge z, y_l \wedge z)$.

Definition 5 *X and Y are said to be **Pareto independent** (resp. **leximin independent, leximax independent**) in the context Z if the plausibility relation $\geq_\Omega$ is Pareto-decomposable (resp. leximin-decomposable, leximax-decomposable) on X and Y in the context Z.*

Proposition 3 *If X and Y are Pareto-independent in the context Z, then they are leximin-independent and leximax-independent. Moreover, all of these relations imply the PO-independence. The converse is not true.*

As it will be discussed in Section 5.2.3, even if a plausibility relation is leximin or leximax decomposable, it cannot be decomposed without loss of information due to the absence of commensurability assumption. Such a problem can be solved by using ranking functions which are introduced now.

4 Background on ranking functions frameworks

In ranking functions frameworks, uncertainty is handled in a qualitative way, but it is encoded on some linearly ordered scale (finite or infinite). Typical examples of these frameworks are possibility theory (Dubois and Prade 1998) where uncertainty is represented in the interval [0, 1] and Spohn's ordinal functions (Spohn

[2] where $max(a,b) \geq_\Omega max(c,d)$ means either [$a \geq_\Omega c$ and $a \geq_\Omega d$] or [$b \geq_\Omega c$ and $b \geq_\Omega d$], and $min(a,b) \leq_\Omega min(c,d)$ means either [$a \leq_\Omega c$ and $a \leq_\Omega d$] or [$b \leq_\Omega c$ and $b \leq_\Omega d$].

1988) which use the set of integers. In the following, we only focus on possibility theory, but results of this paper are also valid for other frameworks such as Spohn's ordinal functions, or Lehmann's ranked models, due to their close relation to possibility theory.

Let us give a brief background on possibility theory (see (Dubois and Prade 1988) for more details). The basic building block is the notion of *possibility distribution* denoted by π and corresponding to a mapping from Ω to $[0, 1]$. A possibility distribution π is said to be *normalized* if it exists at least one world ω which is totally possible, i.e. $\pi(\omega) = 1$.

Given a possibility distribution π, we can define a mapping grading the **possibility** [3] of a formula $\phi \subseteq \Omega$ by: $\Pi(\phi) = max_{\omega \in \phi} \pi(\omega)$.

Each possibility distribution π can generate a plausibility relation (qualitative ordering relation) $\geq_\Omega$ by applying:

$$\omega \geq_\Omega \omega' \text{ iff } \pi(\omega) \geq \pi(\omega'). \quad (5)$$

Similarly, if we define $\geq_\Omega$ from π then:

$$\phi \geq_\Omega \psi \text{ iff } \Pi(\phi) > \Pi(\psi).$$

5 Advantages of the ranking function setting

Let us first analyze the basic differences between ranking functions and the qualitative framework presented in Section 2, from the perspective of the study of independence relations. Basically, there are three differences:

- *Normalization*: in possibility theory, fully plausible worlds receives the grade 1 while there is no counterpart in the qualitative setting.
- *Existence of impossible worlds* graded to 0 in possibility theory, while all worlds are somewhat possible in the qualitative setting.
- *Commensurability* between uncertainty levels, where all rankings reflect grades in the same scale.

In this section, we show the advantages of working with uncertainty ranking functions in order to define more powerful notions of qualitative independence.

[3] A dual measure to the possibility is the necessity degree defined by $N(\phi) = 1 - \Pi(\neg\phi) = min_{\omega \notin \phi}(1 - \pi(\omega))$.

5.1 Consequences of the normalization and of the existence of impossible worlds

5.1.1 Definition of possibilistic conditioning

The normalization and the existence of impossible worlds enables the definition of several notions of conditioning in the graded settings, contrarily to the qualitative setting. This explains why in possibility theory there are two definitions of independence based on conditioning. The natural properties of $\pi' = \pi(. \mid \phi)$ taking into account normalization and impossible worlds become:

$\mathbf{C_1}$: $\forall \omega \notin \phi, \pi'(\omega) = 0$,

$\mathbf{C_2}$: $\forall \omega_1, \omega_2 \in \phi, \pi(\omega_1) > \pi(\omega_2)$ iff $\pi'(\omega_1) > \pi'(\omega_2)$,

$\mathbf{C_3}$: if $\Pi(\phi) = 1$, then $\forall \omega \in \phi, \pi(\omega) = \pi'(\omega)$,

$\mathbf{C_4}$: π' should be normalized,

$\mathbf{C_5}$: if $\pi(\omega) = 0$ then $\pi'(\omega) = 0$.

$\mathbf{C_1}$ confirms that ϕ is a sure piece of information, $\mathbf{C_2}$ says that the new possibility distribution should not affect the possibility degrees relative to the interpretations in ϕ and $\mathbf{C_3}$ says that if ϕ is already consistent with beliefs encoded by π, then the possibility degrees of the elements in ϕ remain identical. $\mathbf{C_5}$ stipulates that impossible worlds remain impossible after conditioning.

The properties ($\mathbf{C_1}$-$\mathbf{C_5}$) do not guarantee a *unique definition of conditioning*. Indeed, the effect of the axiom $\mathbf{C_1}$ may result in a sub-normalized possibility distribution. Restoring the normalization, in order to satisfy $\mathbf{C_4}$, can be done in at least two different ways (when $\Pi(\phi) > 0$) (Dubois and Prade 1988):

- In an ordinal setting, we assign to the best elements of ϕ, the maximal possibility degree (i.e. 1), then we obtain:

$$\pi(\omega \mid_m \phi) = \begin{cases} 1 & if\ \pi(\omega) = \Pi(\phi) \text{ and } \omega \in \phi \\ \pi(\omega) & if\ \pi(\omega) < \Pi(\phi) \text{ and } \omega \in \phi \\ 0 & otherwise \end{cases}$$

  Note that this conditioning form is equivalent to the least specific solution of the combination equation :
  $\pi(\omega \wedge \phi) = min(\pi(\omega \mid_m \phi), \Pi(\phi))$.
  proposed by Hisdal (Hisdal 1978).

- In a numerical setting, we proportionally shift up all elements of ϕ (if the definition makes sense in

the ranking scale):

$$\pi(\omega \mid_P \phi) = \begin{cases} \frac{\pi(\omega)}{\Pi(\phi)} & \text{if } \omega \in \phi \\ 0 & \text{otherwise.} \end{cases}$$

5.1.2 Definitions of possibilistic causal independence

The idea in defining possibilistic causal independence relation based on the possibilistic conditioning is that X is considered as independent from Y in the context Z if for any instance z, the possibility degree of any x remains unchanged for any value y. More formally:

$$\Pi(x \mid y \wedge z) = \Pi(x \mid z), \forall xyz. \quad (6)$$

Since possibility theory has two kinds of conditioning, this leads to two definitions of causal possibilistic independence:

- **Min-based independence relation** is obtained by using $|_m$ in (6); this form of independence is not symmetric (Fonck 1994). If we enforce this property, we get a very strong relation, denoted MS-independence(M for min-based and S for symmetry) since the independence between two sets of variables X and Y implies the ignorance of one of them (De Campos and Huete 1998) i.e. $\pi(x) = 1, \forall x$ or $\pi(y) = 1, \forall y$. In (Ben Amor et al., 2000), it has been shown that:

  Proposition 4 *Let π be a possibility distribution, and $\geq_\Omega$ be its associated plausibility relation. Then X and Y are MS-independent in π if and only if they are Pareto-independent in $\geq_\Omega$.*

- **Product independence relation** obtained by using $|_P$ in (6). This relation is equivalent to:

  $\Pi(x \wedge y \mid_P z) = \Pi(x \mid_P z) * \Pi(y \mid_P z), \forall xyz.$

  The product independence relation, denoted by P-independence, is equivalent to Kappa functions' independence relation based on Spohn-conditioning (Spohn 1988). Note that this relation enjoys the same properties as the independence relation proposed in the probabilistic framework. In particular, we can decompose it, i.e. we can recover $\Pi(x \wedge y \mid_P z)$ in a unique manner from $\Pi(x \mid_P z)$ and $\Pi(y \mid_P z)$ using the product operator.

The following proposition relates possibilistic causal independence to PO-independence:

Proposition 5 *If X and Y are independent in a possibility distribution π according to (6), then they are PO-independent in the plausibility relation induced by π. The converse is false.*

5.2 Effect of the commensurability

In this section we study the *decomposition* of some important independence relations. We will see that the commensurability property is crucial in the recomposition of joint distributions from marginal ones.

5.2.1 Possibilistic Non-Interactivity

In the possibilistic framework, the standard decompositional independence relation between X and Y in the context Z is the **Non-Interactivity (NI)** (Zadeh 1978) defined by:

$$\Pi(x \wedge y \mid_m z) = min(\Pi(x \mid_m z), \Pi(y \mid_m z)), \forall xyz. \quad (7)$$

This relation can be defined in a purely qualitative setting by first defining $\omega \geq_\Omega \omega'$ iff $\pi(\omega) \geq \pi(\omega')$. Then X and Y are NI-independent iff:

$$x \wedge y =_{\Omega|z} x \text{ or } x \wedge y =_{\Omega|z} y, \forall xyz.$$

However, NI-independence is not interesting for qualitative representation since it does not allow the recomposition of a unique global plausibility relation from local orders defined on independent variables (due to the non-satisfaction of the commensurability property), as shown by the example below.

Example : Given two binary variables A and B with the following local orderings:

$$\text{(i) } a_1 >_\Omega a_2$$

$$\text{(ii) } b_2 >_\Omega b_1.$$

There is no unique plausibility relation $\geq_\Omega$ satisfying (i) and (ii) such that A and B are NI-independent. Indeed, it is sufficient to consider the two plausibility relations $\geq_\Omega$ *and* $\geq_{\Omega'}$:

$$a_1 \wedge b_2 >_\Omega a_2 \wedge b_2 >_\Omega a_1 \wedge b_1 =_\Omega a_2 \wedge b_1$$

$$a_1 \wedge b_2 >_{\Omega'} a_2 \wedge b_2 =_{\Omega'} a_1 \wedge b_1 =_{\Omega'} a_2 \wedge b_1$$

However, if the local orderings are encoded in possibility theory then we will have a unique plausibility relation $\geq_\Omega$ using $\pi(a \wedge b) = min(\pi(a), \pi(b)), \forall ab$.

Note that the importance of commensurability assumption also appears in fuzzy set literature, especially in defining connectors between fuzzy sets. For

instance, French (1984) comments the limitation of the definition of the intersection of two fuzzy sets (using the minimum operator to define the membership function associated to the intersection) when no commensurability is assumed.

5.2.2 Decomposition of PO-independence

A natural question now is to see if the encoding of a plausibility relation with a possibility distribution can be useful in the decomposition of the causal independence relation PO. Namely, if there exists a function f such that for each possibility distribution π (encoding some plausibility relation) where X and Y are PO-independent, we have $\forall x, x' \in D_X, \forall y, y' \in D_Y$:

$\pi(x \wedge y) > \pi(x' \wedge y')$ iff $f(\Pi(x), \Pi(y)) > f(\Pi(x'), \Pi(y'))$

Table 1: Possibility distributions on A and B

a	b	$\pi_1(a \wedge b)$	$\pi_2(a \wedge b)$
a_1	b_1	1	1
a_1	b_2	0.9	0.9
a_1	b_3	0.5	0.5
a_2	b_1	0.8	0.8
a_2	b_2	0.3	0.3
a_2	b_3	0.2	0.1
a_3	b_1	0.4	0.4
a_3	b_2	0.1	0.2
a_3	b_3	0	0

In the following, we show that this is impossible in the general case. Indeed, assume that such a function f exists, then let us consider the two possibility distributions π_1 and π_2 defined on two PO-independent variables A and B and given by Table 1.

We have:
$\Pi_1(a_1) = \Pi_2(a_1) = 1, \Pi_1(a_2) = \Pi_2(a_2) = 0.8,$
$\Pi_1(a_3) = \Pi_2(a_3) = 0.4, \Pi_1(b_1) = \Pi_2(b_1) = 1,$
$\Pi_1(b_2) = \Pi_2(b_2) = 0.9, \Pi_1(b_3) = \Pi_2(b_3) = 0.5.$
We can check that A and B are PO-independent in both π_1 and π_2, namely for all a_i, a_j, for all b_k we have: $\pi_l(a_i) > \pi_l(a_j)$ iff $\pi_l(a_i \wedge b_k) > \pi_l(a_j \wedge b_k)$, for l=1,2, and the same exchanging a and b.
Besides, π_1 induces
$\pi_1(a_2 \wedge b_3) > \pi_1(a_3 \wedge b_2)$, i.e. $f(\Pi_1(a_2), \Pi_1(b_3)) > f(\Pi_1(a_3), \Pi_1(b_2))$
while π_2 induces
$\pi_2(a_3 \wedge b_2) > \pi_2(a_2 \wedge b_3)$, i.e. $f(\Pi_2(a_3), \Pi_2(b_2)) > f(\Pi_2(a_2), \Pi_2(b_3))$,
hence a contradiction.

However, there are particular cases where decomposition can be achieved. The first particular case concerns binary variables:

Proposition 6 *If A and B are binary variables, then A and B are PO-independent iff the plausibility relation induced by π is leximin and leximax decomposable.*

The second particular case concerns binary possibility distributions.

Proposition 7 *If π takes its value in a set of two levels (namely the set of interpretations is splitted in only two classes) then X and Y are PO-independent iff the plausibility relation induced by π is leximin and leximax decomposable.*

Such decomposition can be useful for databases when the existing tuples are preferred to absent ones.

The difficulty of decomposing PO-independence relation in the general case is due to the fact that the PO-independence is a weak relation. The following proposition makes the weakness of PO-independence explicit:

Proposition 8 *X and Y are PO-independent in $\geq_\Omega$ iff $\geq_\Omega$ is Pareto-compatible (i.e., monotonic) on X and Y.*

The following subsection discusses the decomposability of Pareto, leximin and leximax relations, which are all Pareto-compatible.

5.2.3 Decomposition of Pareto, leximin and leximax independence

In this subsection we study the decomposition of Pareto, leximin and leximax decomposable relations when we use the possibilistic framework.

The decomposition of leximin and leximax decomposable relations is immediate since we use weights represented by possibility degrees which allows the comparison of different interpretations. We illustrate it on the following example:

Example : Let us consider two variables A and B with the following plausibility relation $\geq_\Omega$ which is leximax decomposable: $a_1 \wedge b_1 >_\Omega a_1 \wedge b_2 >_\Omega a_2 \wedge b_1 >_\Omega a_1 \wedge b_3 >_\Omega a_2 \wedge b_2 >_\Omega a_2 \wedge b_3$

We can easily check that this plausibility relation cannot be recovered from the local orders on A and B induced by it:

(i) $a_1 >_\Omega a_2$

(ii) $b_1 >_\Omega b_2 >_\Omega b_3$

Indeed, it is sufficient to consider the following plausibility relation: $a_1 \wedge b_1 >_{\Omega'} a_2 \wedge b_1 >_{\Omega'} a_1 \wedge b_2 >_{\Omega'} a_1 \wedge b_3 >_{\Omega'} a_2 \wedge b_2 >_{\Omega'} a_2 \wedge b_3$

which satisfies (i) and (ii) and is also leximax decomposable.

However, if $\geq_\Omega$ is encoded by means of a possibility distribution, then the decomposition becomes possible because we preserve the information concerning the relative ordering between any instance of A and any instance of B in the original relation as shown by the following example:

Example : Let π be a possibility distribution encoding the plausibility relation $\geq_\Omega$ given in the previous example (see Table 2).

Table 2: Possibility distribution on A and B

a	b	$\pi(a \wedge b)$
a_1	b_1	1
a_1	b_2	0.9
a_1	b_3	0.5
a_2	b_1	0.8
a_2	b_2	0.3
a_2	b_3	0.2

We can easily recover π from the local distributions on A and B and the numerical scale (1, .9, .8, .5, .3, .2) using the leximax ordering. Indeed, the use of the leximax on the local distributions provides the ordering relation relative to π i.e. $a_1 \wedge b_1 >_\Omega a_1 \wedge b_2 >_\Omega a_2 \wedge b_1 >_\Omega a_1 \wedge b_3 >_\Omega a_2 \wedge b_2 >_\Omega a_2 \wedge b_3$ then using the numerical scale we can recover the original distribution π.

For Pareto-independence, we also have the following result:

Proposition 9 *Let π be a possibility distribution such that X and Y are two Pareto-independent variable sets. Then, π is can be recovered from marginal probabilities $\Pi(X)$ and $\Pi(Y)$ using the product operator, namely:*

$$\pi(x \wedge y) = \Pi(x) * \Pi(y).$$

Note that other operators can also be used to recover the oredering behing π, however the advantage using the product operator is that it preserves both orderings and numerical values.

6 Comparative study

Given a joint possibility distribution π, this section provides a comparative study between the different independence relations presented in this paper. Let $\geq_\Omega$ be the plausibility relation induced from π by using equation 5.

Let us first summarize the different independence relations presented in this paper:

- *Qualitative causal independence:* three definitions have been proposed, depending if we only preserve preferred elements (i.e., I_{PT}), or we only preserve accepted beliefs (i.e., I_{PB}), or we preserve the whole relative ordering (i.e., I_{PO}).
- *Qualitative decomposition independence:* the decomposibility of a plausibility relation has been based on the well-known principles: Pareto-ordering (i.e., I_{Pareto}), Leximin ordering (i.e., $I_{Leximin}$), or leximax ordering (i.e., $I_{Leximax}$).
- *Possibilistic causal independence:* two definitions have been proposed (i.e., I_{MS} and I_P) depending on which of the two definitions of conditioning in possibility theory is used.
- *Possibilistic decomposition independence:* which corresponds to the non-interactivity (i.e., I_{NI}) .

The arrows in Figure 1 show the inclusion of different independence relations. The I_{MS} and I_{Pareto} are the strongest independence relations since the MS or the Pareto independence between two sets of variables implies the ignorance of one of them. However, I_{PT} is the weakest one. Finally, note that I_{NI} is implied by I_{MS} but it is incomparable with the other independence relations.

We give now some counter-examples relative to the non-existent links.

Counter-examples : Let us consider two variables A and B with the possibility distributions given in Table 3.

- with π_1, we can check that the relation $I_{PO}(A, \emptyset, B)$ is true while $I_{MS}(A, \emptyset, B)$, $I_P(A, \emptyset, B)$ and $I_{NI}(A, \emptyset, B)$ are false.
- with π_2 we can check that $I_{NI}(A, \emptyset, B)$ is true contrary to $I_{MS}(A, \emptyset, B)$, $I_{leximin}(A, \emptyset, B)$, $I_{leximax}(A, \emptyset, B)$ $I_P(A, \emptyset, B)$, and $I_{PO}(A, \emptyset, B)$.

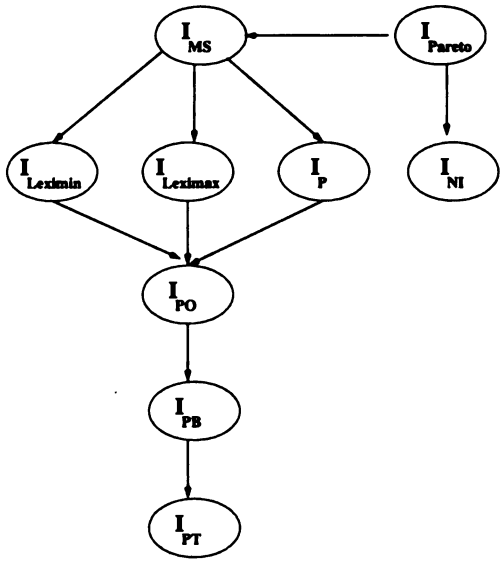


Figure 1: Links between independence relations

Table 3: Possibility distributions on A and B

a	b	$\pi_1(a \wedge b)$	$\pi_2(a \wedge b)$	$\pi_3(a \wedge b)$
a_1	b_1	1	1	0.6
a_1	b_2	0.8	0.8	1
a_2	b_1	0.7	0.8	0.36
a_2	b_2	0.2	0.8	0.6

- with π_3, we can check that the relation $I_P(A, \emptyset, B)$ is true contrary to $I_{MS}(A, \emptyset, B)$ and $I_{NI}(A, \emptyset, B)$.

Suppose now that A is a binary variable and B is a ternary variable and let us consider the possibility distributions given in Table 4.

- we can check that in π_1 $I_{leximax}(A, \emptyset, B)$ is respected while $I_{NI}(A, \emptyset, B)$ is false. In addition, in π_2 $I_{leximin}(A, \emptyset, B)$ is respected contrarily to $I_{NI}(A, \emptyset, B)$.
- with π_1 we can check that A and B are PO-independent but not leximin-independent since $\pi(a_1 \wedge b_3) > \pi(a_2 \wedge b_2)$ while $min(\Pi(a_1), \Pi(b_3)) < min(\Pi(a_2), \Pi(b_2))$.
- with π_2 we can check that A and B are PO-independent but not leximax-independent since $\pi(a_2 \wedge b_2) > \pi(a_1 \wedge b_3)$ while $max(\Pi(a_2), \Pi(b_2)) < max(\Pi(a_1), \Pi(b_3))$.
- with π_1 we can check that A and B are leximax-independent but neither leximin nor Pareto independent. Moreover, with π_2 we can check that A and B are leximin-independent but neither leximax nor Pareto independent.

Table 4: Possibility distributions on A and B

a	b	$\pi_1(a \wedge b)$	$\pi_2(a \wedge b)$
a_1	b_1	1	1
a_1	b_2	0.9	0.9
a_1	b_3	0.5	0.6
a_2	b_1	0.8	0.8
a_2	b_2	0.3	0.7
a_2	b_3	0.2	0.5

- with the following possibility distribution :

a	b	$\pi(a \wedge b)$
a_1	b_1	1
a_1	b_2	0.8
a_2	b_1	0.5
a_2	b_2	0.4
a_3	b_1	0.4
a_3	b_2	0.32

we can check that I_P is satisfied while $I_{leximin}$ and $I_{leximax}$ are false.

Note that in the binary case I_P implies $I_{leximin}$ and $I_{leximax}$.

- Lastly,let A and B be two ternary varibles. In the following possibility distributions, we can check that in π_1 $I_{leximax}$ is respected while I_P is false and that in π_2 $I_{leximin}$ is respected contrarily to I_P.

a	b	$\pi_1(a \wedge b)$	$\pi_2(a \wedge b)$
a_1	b_1	1	1
a_1	b_2	0.9	0.9
a_1	b_3	0.5	0.6
a_2	b_1	0.8	0.8
a_2	b_2	0.3	0.7
a_2	b_3	0.2	0.5

7 Concluding remarks

This paper relates the notions of independence relations defined in purely qualitative setting (when only a total pre-order is used) to the ones defined in ranking function frameworks. Two kinds of independence have been investigated : causal and decomposition ones. A

first point is that independence relations defined in a purely qualitative framework are very weak from the decomposability point of view. This paper has shown that one way to overcome this limitation is to use a ranking framework, like possibility theory. A second point is that the decomposition of a joint distribution in possibility theory is not unique, contrary to probability theory where only the product operator is used. In possibility theory alternative operator, like leximin or leximax, can be used as well. Note that this may be worth applying to other types of numerical distributions (e.g., probability distributions) which will be leximin-independent but not independent in the usual sense of the considered uncertainty theory, in the same way as in (Wong and Butz, 1999) where weak notions of independence are exploited in the probabilistic setting.
A third point is that most of decomposition-oriented forms of independence (except the non-interactivity) are also causal. This clearly appears in Figure 1.

The notions of independence proposed in this paper extend previous works in default reasoning (Benferhat et al. 1994), and belief revision (Dubois et al. 1997b) on independence between events to the case of variables which are not necessarily binary. A line for further research concerns logical counterpart of leximin and leximax independence relations in the possibilistic setting. Indeed, procedures for translating graph representations (defined from min-based and product-based conditional independence) into stratified possibilistic logic bases have been recently exhibited (Benferhat et al. 1999). This is worth doing for leximin-based independence, which is stronger than min-based independence, but still meaningful in a qualitative setting.
Another line of research is to relate the results on the decomposition of a joint distribution defined from independent relations, to the ones provided in multi-criteria decision making for preferential independence.

References

F. Bacchus and A. J. Grove (1996). Utility independence in qualitative decision theory. Proceedings of the 5th Int. Conf. on Principles of Knowledge Representation and Reasoning KR-96: 542-552.

N. Ben Amor, S. Benferhat and K. Mellouli (2000). Possibilistic Independence vs Qualitative Independence. Submitted to IPMU'2000.

S. Benferhat, D. Dubois and H. Prade (1994). Expressing Independence in possibilistic framework and its application to default reasoning. Proceedings of ECAI'94: 150-153.

S. Benferhat, D. Dubois, L. Garcia, and H. Prade (1999). Possibilistic logic bases and possibilistic graphs. Proceedings of the 15th Conf. on Uncertainty in Artificial Intelligence UAI'99.

C. Boutilier, R. I. Brafman, H. H. Hoos and D. Poole (1999). Reasoning with conditional Ceteris Paribus preference statements. Proceedings of the 15th Conf. on Uncertainty in Artificial Intelligence UAI'99.

A. Darwiche (1997), A logical notion of conditional independence: properties and applications. *Artifitial intelligence*: 45-82.

L. M de Campos and J. F. Huete (1998). Independence concepts in possibility theory. *Fuzzy Sets and Systems.*

J. Doyle and M. P. Wellman (1991). Preferential semantics for goals. Proceedings of the National Conf. on Artif. Intellig. AAA'91: 698-703.

D. Dubois, Belief structures (1986). Possibility theory and decomposable confidence measures on finite sets. *Computers and Artificial Intelligence* (Bratislava) **5**: 403-416.

D. Dubois, H. Fargier, H. Prade (1997a) Decision-making under ordinal preferences and comparative uncertainty. Proceedings of the 13th Conf. on Uncertainty in Artificial Intelligence (UAI97) (D. Geiger, P.P. Shenoy, eds.), Providence, RI, Aug. 1-3, 1997, Morgan and Kaufmann, San Francisco, CA, 157-164.

D. Dubois, L. Farinas del Cerro, A. Herzig and H. Prade (1997b). Qualitative relevance and independence: a roadmap. Proceedings of IJCAI'97: 62-67. A long version appears in "Fuzzy sets, logics and reasoning about knowledge", edited by Dubois et al., pp. 325-350, Kluwer Academic Publishers.

D. Dubois and H. Prade (1988). *Possibility theory: An approach to computerized, Processing of uncertainty.* Plenium Press, New York.

D. Dubois, H. Prade (1995). Numerical representation of acceptance. Proceedings of the 11th Conf. on Uncertainty in Artificial Intelligence UAI'95: 149-156.

D. Dubois, H. Prade, P. Smets (1996). Representing Partial Ignorance. IEEE Transaction on Systems, Man and Cybernetics-Part A: Systems and Humans, Vol. 26, N0 03, pp: 361-377.

D. Dubois, H. Prade (1998). Possibility theory: qaulitative and quantitative aspects. In Handbook of defeasible reasoning and uncertainty management systems. Vol. 1, PP. 169-226, Kluwer Academic Press.

S. French (1984).Decision Theory: An introduction to the Mahematics of Rationality, John Wiley and Sons,

New York.

P. Fonck (1994). Conditional independence in possibility theory. *Uncertainty in Artificial Intelligence*: 221-226.

N. Friedman, J. Halpern (1996). Plausible measures and default reasoning. Proceedings of 13th National Conf. on Artif. Intellig., AAAI-96: 1297-1304.

N. Friedman, J. Halpern (1995). Plausibility measures: A user's Guide. Proceedings of the 11th Conf. on Uncertainty in Artificial Intelligence UAI'95: 175-184.

J. Halpern (1997). Defining relative likelihood in partially-ordered preferential structure. *Journal of Artificial Intelligence Research* **7**: 1-24.

E. Hisdal (1978). Conditional possibilities independence and non interaction. *Fuzzy sets and Systems* **1**.

G. Lakemeyer (1997). Relevance from an epistemic perspective. *Artificial Intelligence*: 137-167.

J. Lang, P. Marquis (1998). Complexity results for independence and definability in propositional logic. Proceedings of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning KR'98: 356-367.

D. Lehmann (1989). What does a conditional knowledge base entail?. Proceedings KR'89: 357-367.

H. Moulin (1988). Axioms for cooperative decision-making, Cambridge University Press, Cambridge.

W. Spohn (1988). Ordinal conditional functions: a dynamic theory of epistemic states Causation in decision. *Belief changes and statistics.* W. Harper and B. Skyrms (ed.): 105-134.

S. K. M. Wong, C. J. Butz, and Y. Xiang(1995). A method for implementing a probabilistic model as a relational database. Proceedings of the 11th Conf. on Uncertainty in Artificial Intelligence UAI-95: 556-564.

S. K. M. Wong, C. J. Butz (1999). Contextual weak independence in bayesian networks. Proceedings of UAI-99: 670-679.

L. A. Zadeh (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems* **1**: 3-28.

Spatial representation of spatial relationship knowledge

Isabelle Bloch
Ecole Nationale Supérieure des Télécommunications
Département TSI - CNRS URA 820
46 rue Barrault, 75013 Paris, France
Isabelle.Bloch@enst.fr

Abstract

Spatial relationships between objects are very important for tasks such as spatial reasoning, scene interpretation, object recognition in various applications, since they carry structural information about the spatial arrangement of objects in the space. Such spatial relationships can be modeled and represented in different formalisms, most of them aiming at assessing or computing relationships between given objects. Here we take another point of view, and use knowledge about relationships as constraints for exploring the space, reducing the search area of an object, and progressive spatial reasoning. This calls for a different type of representation. We propose here spatial representations of such knowledge, in a fuzzy set framework, that allows for representation of quantitative, qualitative, imprecise, heterogeneous knowledge. In this paper, we detail representation of knowledge about set relationships, adjacency, distances, directional relative position in the 3D space, in a unified way through mathematical morphology.

1 INTRODUCTION

The interest of relationships between objects has been highlighted in very different types of works: in vision, for identifying shapes and objects, in database system management, for supporting spatial data and queries, in artificial intelligence, for planning and reasoning about spatial properties of objects, in cognitive and perceptual psychology, in geography, for geographic information systems.

Most works aim at defining and assessing spatial relationships between objects, given these objects. For instance in model-based structural pattern recognition, the aim is to recognize objects from the comparison between the characteristics of objects in the scene and objects in the model, and from comparison between relationships of pairs of objects (or more) in the scene and in the model. This assumes that regions are given, and have only to be identified.

Here we take another point of view and address the problem of the representation of knowledge about expected relationships, in order to guide the reasoning process in the space, for exploring the space and search for the object that satisfies some relationships with respect to already known objects. For the example of model-based pattern recognition, this leads to progressive recognition, where each object is detected and recognized by gathering constraints given by the model expressing relationships that this object has to satisfy with respect to previously recognized objects. The model used may be heterogeneous. Typically, it may have two distinct parts: one iconic part, where each (crisp or fuzzy) region has a unique label is an object or a structure (this is typically the case for digital maps as used in aerial and satellite imaging, or 3D anatomical atlas as used in medical imaging, or environment maps as used in robotics), and a propositional part, expressing expert knowledge, as linguistic terms, logical propositions, possibly including numerical, qualitative or imprecise values.

Therefore, the requirements for the representation of spatial relationships are to allow the definition of spatial constraints and exploration of space, to combine heterogeneous knowledge, and to cope with imprecision. This calls for a common framework. We propose here to use directly the spatial domain as representation space, and fuzzy sets as imprecision management framework. Each relationship is expressed as one spatial fuzzy set, corresponding to a spatial constraint, restricting the space to the

only regions where the relationship is satisfied. Although several works in image processing, robotics, etc. make use of spatial fuzzy sets to represent objects, to our knowledge, very few such representations have been proposed previously for relationships. In [Koczy(1988)], fuzzy areas are defined for representing directional relative position, but only on one axis, on which projections of objects are considered. In [Gasós and Ralescu(1997), Gasós and Saffiotti(2000)], for applications in robotics, lines are represented as spatial fuzzy sets to account for uncertainty, and distances between objects expressed as linguistic variables are represented as fuzzy sets on each axis. Here we propose spatial representations in the same space as the objects themselves.

The applications that can be anticipated from the proposed representation concern spatial reasoning, answering queries, finding and recognizing objects, spatial indexing, information retrieval, in various domains, such as aerial, satellite and medical imaging, robotics, GIS, video, vision.

In Section 2, we present the main idea of representing knowledge on spatial relationships as spatial fuzzy sets. The main features of the proposed approach are (i) to represent spatial knowledge as constraints in the space where relationships are satisfied, instead of binary or n-ary relations, (ii) to use this knowledge representation to explore the space, and progressively recognize objects and interpret a scene, (iii) to have a unified approach of many spatial concepts via mathematical morphology, (iv) to represent imprecision in knowledge via spatial fuzzy sets and fuzzy mathematical morphology. We detail the proposed formalism for several types of knowledge, about object in Section 3 (only the example of shape and localization is presented), about topological relationships in Section 4, and about metric relationships in Section 5. We illustrate the proposed approach in Section 6 with the example of brain structures.

2 SPATIAL FUZZY SETS AS A REPRESENTATION FRAMEWORK

Usually vision and image processing make use of quantitative representations of spatial relationships. In artificial intelligence, mainly symbolic representations are developed (see [Vieu(1997)] for a survey). Limitations of purely qualitative reasoning has already been stressed in [Dutta(1991)], as well as the interest of adding semiquantitative extension to qualitative value (as done in the fuzzy set theory for linguistic variables [Zadeh(1975), Dubois and Prade(1980)]) for deriving useful and practical conclusions (as for recognition). Purely quantitative representations are limited in the case of imprecise statements, and of knowledge expressed in linguistic terms. Here we propose to integrate both quantitative and qualitative knowledge, using semiquantitative interpretation of fuzzy sets. As already mentioned in [Freeman(1975)], this allows to provide a computational representation and interpretation of imprecise spatial constraints, expressed in a linguistic way, possibly including quantitative knowledge.

We consider the general case of a 3D space, where objects can have any shape and any topology. According to the semantical hierarchy proposed in [Kuipers and Levitt(1988)], we consider topological and metric relationships (corresponding to levels 3 and 4 of this hierarchy). Many authors have stressed the importance of topological relationships, e.g. [Allen(1983), Varzi(1996), Randell et al.(1992), Cohn et al.(1997), Asher and Vieu(1995), Clementini and Felice(1997), Kuipers(1978), Pullar and Egenhofer(1988)]. But distances and directional relative position (constituting the metric relationships) are also important, e.g. [Peuquet(1988), Dutta(1991), Kuipers and Levitt(1988), Gapp(1994), Krishnapuram et al.(1993), Wang and Keller(1999), Liu(1998)].

Knowledge used for guiding recognition of an object or to perform spatial reasoning is generally heterogeneous. It may concern the object we are looking at (its shape, topology, color, position), or relationships to other objects (distances, adjacency, relative directional position). It may be generic (typically if derived from a model or from expert knowledge), or factual (if derived from the scene itself). And it may be usually provided in a lot of different forms. Classically it can be a number, a distribution or a binary value. But we will also be concerned with imprecise values and with propositional formulas which are often used by experts within a given application. Imprecise values are expressed sometimes in linguistic terms: for instance the expected distance between two objects (*close*, *far*, ...). They can also be expressed as an interval. Propositional formulas (*object A is to the right of object B*) usually express rather complex ideas which need much prior knowledge to be correctly interpreted.

The main idea of this paper is to translate all available knowledge as a spatial representation. Such representations can then be used in a fusion process that combines all these regions of interest in order to focus attention by reducing the search space and to

restrict it to the area that satisfies most relationships. Since many pieces of information are delivered in an imprecise way, we make use of the framework of fuzzy sets [Zadeh(1965), Dubois and Prade(1980)]. This modeling is well adapted to information derived from a model for instance, since models contain generic knowledge, which is usually selectively simplified and schematized to bring the essential information to the fore, and therefore subject to imprecision. Also the scene to be processed, or the space where reasoning has to be carried out, usually suffers from imprecision, due to the observed phenomenon itself or to the way it is observed. In this context, the theory of fuzzy sets appears to be well suited. Indeed, it provides a good theoretical basis to model the imprecision of the information at different levels of representation, and to represent both numerical and symbolic information, including structural information (constituted mainly by spatial relationships here). Moreover, fuzzy set theory has benefited from the many recent developments in information fusion, in the definitions of combination operators, of similarity measures, and in decision tools, that can be useful to manipulate the proposed representations. An illustration of this for structural model-based pattern recognition is described in [Géraud et al.(1999)].

In the following, the space to be processed will be denoted by $\mathcal{S}$ (we consider here the case of 3D discrete space), and a point (volume element or voxel) in $\mathcal{S}$, by v. For each piece of knowledge, we consider its "natural expression", i.e. the usual form in which it is given or available, and translate it into a spatial fuzzy set in the space, the membership of which is denoted by $\mu_{\text{knowledge}}$:

$$\mu_{\text{knowledge}} : \begin{cases} \mathcal{S} & \to & [0,1] \\ v & \mapsto & \mu_{\text{knowledge}}(v). \end{cases} \quad (1)$$

In this representation, each piece of knowledge becomes a fuzzy region of the space. If the knowledge is considered as a constraint to be satisfied by the object to be recognized, this fuzzy region represents a search area or a fuzzy volume of interest for this object.

This type of representation provides a common framework to represent pieces of information of various types (objects, spatial imprecision, relationships to other objects, etc.). Therefore the fuzzy regions defined in the space $\mathcal{S}$ corresponding to these pieces of information may have different semantics. Moreover, this common framework allows the combination of this heterogeneous information.

The numerical representation of membership values assumes that we can assign numbers that represent degrees of satisfaction of a relationship for instance. These numbers can be derived from prior knowledge or learned from examples, but usually there remain some quite arbitrary choices. This might appear as a drawback in comparison to propositional representations. However, it is not necessary to have precise estimations of these values, and experimentally we observed a good robustness with respect to these estimations, in various problems like information fusion, object recognition and scene interpretation [Bloch et al.(1996), Géraud et al.(1999)]. This can be explained by two reasons: first, the fuzzy representations are used for rough information and therefore do not have to be precise itself, and second several pieces of information are usually combined in a whole reasoning process, which decreases the influence of each particular value (of individual information). Therefore the chosen numbers are not crucial. What is important is that ranking is preserved. For instance if a region of the space satisfies a relationship to some objects to a higher degree that another region, then this ranking is preserved in the representation, for all relationships described in the following sections. Assuming the existence of ranking is reasonable for the type of relations we consider.

3 SHAPE AND LOCALIZATION

Knowledge about shape and localization of an object can be provided by a rough iconic description of the object, as in the case of a map or an anatomical atlas. Such knowledge is subject to imprecision due to the schematic aspect of the model, the variability between instances of the model, and imperfect correspondence between model and scene. The spatial representation of this knowledge in the space $\mathcal{S}$ has to take this imprecision into account.

We propose to do this by extending the region given by the model in a fuzzy way, in order to take into account these imprecisions. An appropriate tool for this is fuzzy morphological dilation, defined as [Bloch and Maître(1995)]:

$$\forall v \in \mathcal{S},\ D_\nu(\mu)(v) = \sup_{v' \in \mathcal{S}} t[\mu(v'), \nu(v' - v)], \quad (2)$$

where μ denotes the object to be dilated (typically a model object), ν denotes a fuzzy set (also defined in $\mathcal{S}$) called structuring element, $D_\nu(\mu)$ denotes the dilation of μ by ν, and t is a t-norm[1]. Other forms of dilation are possible (see [Bloch and Maître(1995)] for a

[1] A triangular norm (or t-norm) is a function from $[0,1] \times [0,1]$ into $[0,1]$ which is commutative, associative, increasing, and for which 1 is unit element and 0 is null element [Menger(1942),

review, with a comparison of properties), corresponding to different expressions of the degree of intersection between μ and the translation of ν at each possible position in $\mathcal{S}$. Equation 2 applies for both crisp and fuzzy objects and structuring elements. In this Equation, a fuzzy object, defined through its membership function μ from $\mathcal{S}$ into $[0,1]$, is transformed by means of another fuzzy object, defined through its membership function ν also from $\mathcal{S}$ into $[0,1]$ and translated at each position in $\mathcal{S}$. This transformation provides also a fuzzy object, the membership function of which is a function $D_\nu(\mu)$ from $\mathcal{S}$ into $[0,1]$.

Figure 1 illustrates an example of a set of objects (buildings extracted from a map), and of dilation of one of these objects by a fuzzy structuring element. The structuring element can be interpreted as imprecision in the drawing of the building in the map, and allows to define an area where the corresponding building can be searched in an aerial image for instance.

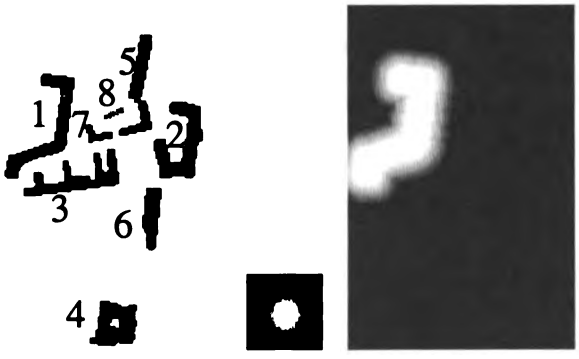


Figure 1: Left: buildings extracted from a map. Middle: fuzzy structuring element. Right: fuzzy dilation of building 1. Membership values vary from 0 (black) to 1 (white).

Representing imprecise objects as spatial fuzzy sets has been used in several works (in image processing or in robotics for instance). The originality of the proposed representation is that it provides a formal and consistent way to introduce imprecision represented as a fuzzy set ν, while preserving some morphological information about the object (see Figure 1). One of the advantage of this representation comes from the algebraic properties of dilation.

Indeed, fuzzy dilation satisfies a set of properties, some of which are important (and even requirements) for its use here:

- It is extensive if $\nu(O) = 1$, where O denotes the center of the structuring element:

$$\mu \subset D_\nu(\mu), \text{ i.e. } \forall v \in \mathcal{S},\ D_\nu(\mu)(v) \geq \mu(v), \quad (3)$$

which guarantees that the shape is actually extended in order to account for imprecision and variability (fuzzy set inclusion is defined in a classical way using $\leq$ on membership functions).

- It is increasing with respect to both the structuring element and the set to be dilated:

$$\nu \subset \nu' \Rightarrow D_\nu(\mu) \subset D_{\nu'}(\mu), \quad (4)$$

$$\mu \subset \mu' \Rightarrow D_\nu(\mu) \subset D_\nu(\mu'). \quad (5)$$

This property guarantees that the larger the structuring element, the more imprecision is introduced, and that the larger the object in the model, the larger the volume of interest in $\mathcal{S}$ (in all these equations, subsethood of fuzzy sets is defined as above, using $\leq$);

- Dilation satisfies an iteration property:

$$D_{\nu'}[D_\nu(\mu)] = D_{\nu' \oplus \nu}(\mu), \quad (6)$$

which allows to express different sources of imprecision by different structuring elements, and to introduce these imprecisions either sequentially, or globally by using the Minkowski addition ($\oplus$) of all structuring elements.

- Dilation commutes with union, which guarantees that objects can be processed indifferently globally or by parts:

$$D_\nu(\mu \cup \mu') = D_\nu(\mu) \cup D_\nu(\mu'), \quad (7)$$

where union of fuzzy sets is defined as the maximum of membership functions, i.e. $\forall v \in \mathcal{S}$:

$$D_\nu(\max(\mu(v), \mu'(v))) = \max(D_\nu(\mu)(v), D_\nu(\mu')(v)).$$

The important choice to be made here concerns the structuring element which represents the spatial imprecision. When all the existing sources of imprecision are taken into account into the fuzzy volume of interest, this volume should contain the object we are looking at. The choice of the structuring element reflecting the possible imprecisions depends on the application at hand. Its extent can be defined from prior knowledge on the type of imprecision (see e.g. [Bloch et al.(1996)]), or learned from a set of representative instantiations of the model. Also the method proposed in [Gasós and Saffiotti(2000)] for deriving uncertainty and membership degrees to line segments from scatter information issued from observations could be used for defining structuring elements. Then dilation appears as a formal way to introduce this scatter information in the definition of lines.

Schweizer and Sklar(1963)]. Examples of t-norms are min, product, etc. [Dubois and Prade(1980)].

4 TOPOLOGICAL RELATIONSHIPS BETWEEN OBJECTS

Topological relationships between objects correspond to the third level of the semantical hierarchy of Kuipers [Kuipers and Levitt(1988)], and include part-whole relationships (inclusion, exclusion, adjacency, etc.).

4.1 SET RELATIONSHIPS

Set relationships specify if areas where other objects can be localized are forbidden or obligatory. This goes with the idea of progressively exploring the space. These set relationships are expressed as inclusion in objects (denoted by O^{in}) or exclusion from objects (denoted by O^{out}). The corresponding region of interest has the following membership function:

$$\forall v \in \mathcal{S},\ \mu_{\text{set}}(v) = \begin{cases} 1 & \text{if } v \in O^{\text{in}} \setminus O^{\text{out}} \\ 0 & \text{elsewhere.} \end{cases} \quad (8)$$

If O^{in} and O^{out} are fuzzy objects, defined on $\mathcal{S}$ through their membership functions $\mu_{O^{\text{in}}}$ and $\mu_{O^{\text{out}}}$, then the previous equation becomes:

$$\mu_{\text{set}}(v) = t[\mu_{O^{\text{in}}}(v), 1 - \mu_{O^{\text{out}}}(v)], \quad (9)$$

where t is a t-norm, which expresses a conjunction between inclusion constraint and exclusion constraint. The properties of t-norms guarantee that intuitive requirements are satisfied:

- Since any t-norm is smaller than the min, μ_{set} is included in $\mu_{O^{\text{in}}}$, i.e.: $\forall v \in \mathcal{S},\ \mu_{\text{set}}(v) \leq \mu_{O^{\text{in}}}(v)$.
- Similarly, μ_{set} is included in the complement of $\mu_{O^{\text{out}}}$, i.e.: $\forall v \in \mathcal{S},\ \mu_{\text{set}}(v) \leq 1 - \mu_{O^{\text{out}}}(v)$.
- Increasingness of t induces increasingness of μ_{set} with respect to $\mu_{O^{\text{in}}}$ and decreasingness with respect to $\mu_{O^{\text{out}}}$, which expresses that constraining a fuzzy region to be included in smaller objects (respectively excluded from larger objects) leads to a smaller fuzzy region.
- Commutativity and associativity of t-norms allow to introduce constraints on inclusion in several objects (or exclusion from several objects) in any order. For instance: $t[t[\mu_{O^{\text{in}}}(v), 1 - \mu_{O^{\text{out}}}(v)], \mu_{O'^{\text{in}}}(v)] = t[t[\mu_{O'^{\text{in}}}(v), 1 - \mu_{O^{\text{out}}}(v)], \mu_{O^{\text{in}}}(v)]$.

An example is shown in Figure 7.

4.2 ADJACENCY

Adjacency expresses an often used topological relation in structural recognition. Only a few attempts in the literature address the problem of fuzzy adjacency. Fuzzy topology was introduced in [Rosenfeld(1979)], where a fuzzy connectivity between points is defined but without reference to the notion of fuzzy neighborhood, or to fuzzy adjacency. Similar approaches can also be found in [Rosenfeld(1984), Udupa and Samarasekera(1996)], where degrees of connectivity in a fuzzy set are also introduced, but neither the connectivity nor the adjacency between two fuzzy sets are defined. Rosenfeld and Klette [Rosenfeld and Klette(1985)] define a degree of adjacency between two crisp sets, using a geometrical approach based on the notion of "visibility" of a set from another one. This definition is then extended to degree of adjacency between two fuzzy sets. However, this definition is not symmetrical, and probably not easy to transpose to higher dimensions. In [Bloch et al.(1997)] we proposed several definitions for degree of adjacency between fuzzy sets coping with spatial imprecision. Here we extend this work for defining fuzzy regions expressing an adjacency constraint with an object.

As shown in [Bloch et al.(1997)], adjacency can be directly related to distance, since two sets A and B are adjacent if and only if they are not intersecting and their minimum distance is equal to 0 or 1 (depending on the conventions chosen in the discrete space). The extension to the fuzzy case is based on fuzzy dilation. Therefore, the same formalism as the one developed for distances can be used for this relationship (see Section 5), which exhibits a link between topological and metric relationships.

More directly, fuzzy dilation can be used to define the external boundary of a fuzzy object μ, as:

$$b_e(\mu) = t[D_\nu(\mu), 1 - \mu]. \quad (10)$$

In a similar way, internal boundary is defined from fuzzy erosion, as:

$$b_i(\mu) = t[\mu, D_\nu(1 - \mu)] = t[\mu, 1 - E_\nu(\mu)], \quad (11)$$

where $E_\nu(\mu)$ denotes the fuzzy erosion of μ by ν [Bloch and Maître(1995)]:

$$\forall v \in \mathcal{S},\ E_\nu(\mu)(v) = \inf_{v' \in \mathcal{S}} T[\mu(v'), 1 - \nu(v' - v)], \quad (12)$$

where T is a t-conorm[2].

[2] A triangular conorm (or t-conorm) is a function from

The search area for an object adjacent to μ is then expressed by two fuzzy sets: $b_e(\mu)$ that should be intersected by the boundary of the searched object, and $1-\mu$ which expresses the inclusion constraint in the complement of μ.

From these basic topological relationships (inclusion, exclusion, adjacency), other ones can be derived. For instance, an object that is a tangential proper part of μ has to be searched in μ and its boundary has to intersect $b_i(\mu)$. A non tangential proper part of μ has to be searched in $E_\nu(\mu)$.

5 METRIC RELATIONSHIPS BETWEEN OBJECTS

According to the fourth level of the semantical hierarchy of Kuipers [Kuipers and Levitt(1988)], metric relationships include distances and directional relative position.

5.1 DISTANCES

Distance between objects is an important information for the assessment of spatial arrangement between objects in a scene. Therefore they are widely used in structural pattern recognition and spatial reasoning. Distances between objects A and B can be expressed in different forms, as *the distance between A and B is equal to n, the distance between A and B is less (respectively greater) than n, the distance between A and B is between n_1 and n_2*. In the framework of our study, these expressions will be translated in spatial volumes of interest within $\mathcal{S}$, taking into account imprecision and uncertainty, since these statements are generally approximate (n, n_1, n_2 can be numbers, but also intervals, fuzzy numbers, linguistic values, etc.).

In the literature on fuzzy distances, we distinguish on the one hand distances that basically compare only the membership functions representing the concerned fuzzy objects, and, on the other hand, distances that combine spatial distance between objects and membership functions (see e.g. [Zwick et al.(1987), Bloch(1999b)] for a review). The second class of methods finds more general applications in spatial representation and spatial reasoning since they take into account both spatial information and information related to the imprecision attached to the objects. New distances based on mathematical morphology are proposed in this second class in [Bloch(1999b)]. Here we extend this work and show how to build fuzzy regions that represent some distance relationships to an object. In contrary to the approach proposed in [Gasós and Ralescu(1997), Gasós and Saffiotti(2000)] where linguistic variables about distances are represented as fuzzy sets on each axis, from which distance knowledge in the space can be derived, we choose here to represent distance knowledge directly in the space $\mathcal{S}$, as spatial fuzzy sets. The method we propose is independent of the dimension of $\mathcal{S}$.

Distances between sets (average, Hausdorff, minimum distances) are usually defined by analytical expressions. But they also have equivalents in set theoretical terms by means of mathematical morphology. This allows to easily include imprecision, and to deal with distances between fuzzy sets and with fuzzy distances [Bloch(1999b)]. Let us detail these equivalences.

We first consider the crisp case, and the minimum distance (other distances could be used as well) in a bounded discrete space. Let $d(A,B)$ be the minimum distance between two crisp sets A and B, i.e.:

$$d(A,B) = \min_{v\in A, v'\in B} d_E(v,v'), \tag{13}$$

where $d_E(v,v')$ denotes the underlying distance between points of $\mathcal{S}$ (Euclidean distance or a discrete distance). Let B^n be a ball of radius n, according to the distance d_E. For simplifying the notations, we set: $D^n(A) = D_{B^n}(A)$, which is the dilation of size n of A (i.e. the dilation with a ball of size n as the structuring element).

The following equations hold:

$$D^n(A) = \{v \in \mathcal{S}, \min_{v'\in A} d_E(v,v') \leq n\}. \tag{14}$$

$$d(A,B) = \min\{n \in \mathbb{N},\ D^n(A)\cap B \neq \emptyset \text{ and } D^n(B)\cap A \neq \emptyset\}. \tag{15}$$

$$d(A,B) = n \Leftrightarrow$$
$$\begin{cases} \forall m < n, D^m(A)\cap B = D^m(B)\cap A = \emptyset \\ \text{and } D^n(A)\cap B \neq \emptyset, D^n(B)\cap A \neq \emptyset. \end{cases} \tag{16}$$

$$d(A,B) \leq n \Leftrightarrow D^n(A)\cap B \neq \emptyset, D^n(B)\cap A \neq \emptyset. \tag{17}$$

$$d(A,B) \geq n \Leftrightarrow \forall m < n, D^m(A)\cap B = D^m(B)\cap A = \emptyset. \tag{18}$$

$$n_1 \leq d(A,B) \leq n_2 \Leftrightarrow$$
$$\begin{cases} \forall m < n_1, D^m(A)\cap B = D^m(B)\cap A = \emptyset \\ \text{and } D^{n_2}(A)\cap B \neq \emptyset, D^{n_2}(B)\cap A \neq \emptyset. \end{cases} \tag{19}$$

The proof of these equations involves extensivity of dilation (for such structuring elements), and increasingness with respect to the structuring element.

$[0,1] \times [0,1]$ into $[0,1]$ which is commutative, associative, increasing, and for which 0 is unit element and 1 is null element [Menger(1942), Schweizer and Sklar(1963)]. Examples of t-conorms are max, algebraic sum, etc. [Dubois and Prade(1980)]. They are dual of t-norms with respect to complementation, i.e. $T(1-a,1-b) = 1-t(a,b)$.

We assume that A is known as one already recognized object, or a known area of $\mathcal{S}$, and that we want to determine B, subject to satisfy some distance relationship with A. According to the previous equations, dilation of A is an adequate tool for this. Let us consider the following different cases:

- If knowledge expresses that $d(A,B) = n$, then the border of B should intersect the region defined by $D^n(A) \setminus D^{n-1}(A)$, which is made up of the points exactly at distance n from A, and B should be looked for in $D^{n-1}(A)^C$ (the complement of the dilation of size $n-1$).
- If knowledge expresses that $d(A,B) \leq n$, then B should be looked for in A^C, with the constraints that at least one point of B belongs to $D^n(A) \setminus A$.
- If knowledge expresses that $d(A,B) \geq n$, then B should be looked for in $D^{n-1}(A)^C$.
- If knowledge expresses that $n_1 \leq d(A,B) \leq n_2$, then B should be searched in $D^{n_1-1}(A)^C$ with the constraint that at least one point of B belongs to $D^{n_2}(A) \setminus D^{n_1-1}(A)$.

The constraints on the border lead to the definition of actually two fuzzy sets, one for constraining the object, and one constraining its border, as for adjacency. However, they can be avoided by considering both minimum and maximum (Hausdorff) distances, expressing for instance that B should lay between a distance n_1 and a distance n_2 of A. Therefore, the minimum distance should be greater than n_1 and the maximum distance should be less than n_2. In this case, the volume of interest for B is reduced to $D^{n_2}(A) \setminus D^{n_1-1}(A)$.

In cases where imprecision has to be taken into account, fuzzy dilations are used, with the corresponding equivalences with fuzzy distances [Bloch and Maître(1995), Bloch(1999b)]. The extension to approximate distances calls for fuzzy structuring elements. We define these structuring elements through their membership function ν on $\mathcal{S}$. Structuring elements with a spherical symmetry can typically be used, where the membership degree only depends on the distance to the center of the structuring element.

Let us consider the generalization to the fuzzy case of the last case (minimum distance of at least n_1 and maximum distance of at most n_2 to a fuzzy set μ). Instead of defining an interval $[n_1, n_2]$, we consider a fuzzy interval, defined as a fuzzy set on $\mathbb{R}^+$ having a core equal to the interval $[n_1, n_2]$. The membership function μ_n is increasing between 0 and n_1 and decreasing after n_2. Then we define two structuring elements, as:

$$\nu_1(v) = \begin{cases} 1 - \mu_n(d_E(v,0)) & \text{if } d_E(v,0) \leq n_1 \\ 0 & \text{else} \end{cases} \quad (20)$$

$$\nu_2(v) = \begin{cases} 1 & \text{if } d_E(v,0) \leq n_2 \\ \mu_n(d_E(v,0)) & \text{else} \end{cases} \quad (21)$$

where d_E is the Euclidean distance in $\mathcal{S}$ and O the origin. The spatial fuzzy set expressing the approximate relationship about distance to μ is then defined as:

$$\mu_{\text{distance}} = t[D_{\nu_2}(\mu), 1 - D_{\nu_1}(\mu)] \quad (22)$$

if $n_1 \neq 0$, and $\mu_{\text{distance}} = D_{\nu_2}(\mu)$ if $n_1 = 0$. The increasingness of fuzzy dilation with respect to both the set to be dilated and the structuring element [Bloch and Maître(1995)] guarantees that these expressions do not lead to inconsistencies. Indeed, we have $\nu_1 \subset \nu_2$, $\nu_1(0) = \nu_2(0) = 1$, and therefore $\mu \subset D_{\nu_1}(\mu) \subset D_{\nu_2}(\mu)$. In the case where $n_1 = 0$, we do not have $\nu_1(0) = 1$ any longer, but in this case, only the dilation by ν_2 is considered. This case corresponds actually to a distance to μ less than about n_2. These properties are indeed expected for representations of distance knowledge.

Figure 2 illustrates this approach. The two structuring elements ν_1 and ν_2 are derived from a fuzzy interval μ_n, are used for dilation of an object of Figure 1, and μ_{distance} is computed to represent the approximate knowledge about the distance to this object.

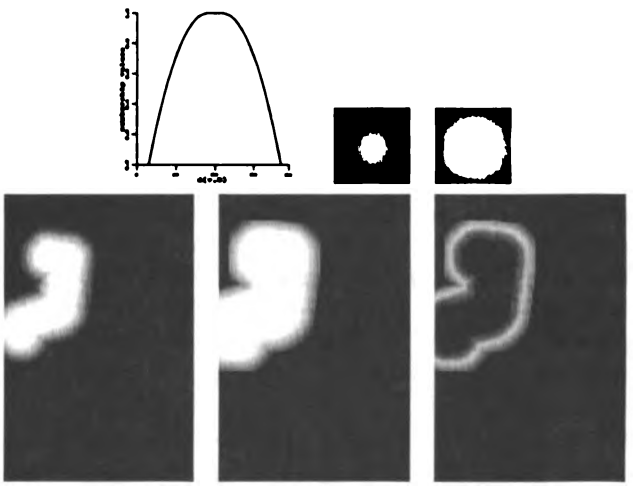


Figure 2: Membership function μ_n, structuring elements ν_1 and ν_2, dilation of building 1 of Figure 1 with these two structuring elements, and representation of μ_{distance}.

From an algorithmic point of view, fuzzy dilations may be quite heavy if the structuring element has a large

support. However, in the case of crisp objects and structuring elements with spherical symmetry, fast algorithms can be implemented. The distance to the object A is first computed using chamfer algorithms [Borgefors(1996)]. It defines a distance map in $\mathcal{S}$, which gives the distance of each voxel v to object A. This discrete distance can be made as precise as necessary [Mangin et al.(1994)]. Then the translation into a fuzzy volume of interest is made according to a simple look-up table derived from μ_n. This algorithm has a linear complexity in the cardinality of $\mathcal{S}$.

5.2 RELATIVE DIRECTIONAL POSITION

In contrary to the previous relationships, relative directional position (like *object A is on the right of object B*) is an intrinsically vague information. The fuzzy set framework is appropriate to define formally such relationships with good properties.

To the best of our knowledge, almost all existing methods for defining fuzzy relative directional spatial position rely on angle measurements between points of the two objects of interest [Krishnapuram et al.(1993), Miyajima and Ralescu(1994), Keller and Wang(1995), Matsakis and Wendling(1999)], and concern 2D objects (sometimes with possible extension to 3D). These approaches cannot easily be used for defining a fuzzy set in $\mathcal{S}$ corresponding to the area where a directional relationship to an object is satisfied.

We proposed in [Bloch(1999a)] a different approach, which is more suitable to this task, since the relationship is defined directly in $\mathcal{S}$. It is based on a morphological approach. Let us consider a reference object A and an object B for which the relative position with respect to A has to be evaluated. In order to evaluate the degree to which B is in some direction with respect to A, we use a two-step method. We first define a spatial fuzzy set around the reference object A as a fuzzy set such that the membership value of each point corresponds to the degree of satisfaction of the spatial relation under examination. The fuzzy set is defined in the same space as the considered objects, in the contrary to the solution proposed in [Koczy(1988)], where the fuzzy area is defined on a one-dimensional axis, from a projection of the object. Then we compare the object B to the spatial fuzzy set attached to A, in order to evaluate how well this object matches with the areas having high membership values (i.e. areas that are in the desired direction). This evaluation is done using a fuzzy pattern matching approach [Dubois and Prade(1988)], which provides as a result an interval (and not a single number).

For the application here described, the first step only is needed, which provides directly the spatial fuzzy set we are interested in. This step is detailed below in a 3D space.

In the 3D Euclidean space, a direction is defined by two angles α_1 and α_2, with $\alpha_1 \in [0, 2\pi]$ and $\alpha_2 \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ ($\alpha_2 = 0$ in the 2D case). We note $\alpha = (\alpha_1, \alpha_2)$. The direction of interest is denoted by $\vec{u}_{\alpha_1,\alpha_2} = (\cos\alpha_2 \cos\alpha_1, \cos\alpha_2 \sin\alpha_1, \sin\alpha_2)^t$. We denote by $\mu_\alpha(A)$ the fuzzy region representing the relation *to be in the direction* $\vec{u}_{\alpha_1,\alpha_2}$ *with respect to reference object A*. Points that satisfy this relation with high degrees should have high membership values. In other terms, the membership function $\mu_\alpha(A)$ has to be an increasing function of the degree of satisfaction of the relation. The requirements stated above for this fuzzy set are not strong and leave room for a large spectrum of possibilities. We propose a definition that looks precisely at the domains of space that are visible from a reference object point in the direction $\vec{u}_{\alpha_1,\alpha_2}$. This applies to objects of any kind, in particular having strong concavities.

Let us denote by P any point in $\mathcal{S}$, and by Q any point in A. Let $\beta(P,Q)$ be the angle between the vector $\vec{QP}$ and the direction $\vec{u}_{\alpha_1,\alpha_2}$, computed in $[0, \pi]$:

$$\beta(P,Q) = \arccos\left[\frac{\vec{QP} \cdot \vec{u}_{\alpha_1,\alpha_2}}{\|\vec{QP}\|}\right], \text{ and } \beta(P,P) = 0. \tag{23}$$

Setting $\beta(P,P) = 0$ allows actually to deal with overlapping objects or with fuzzy objects with overlapping supports.

We then determine for each point P the point Q of A leading to the smallest angle β, denoted by $\beta_{\min}$. In the crisp case, this point Q is the reference object point from which P is visible in the direction the closest to $\vec{u}_{\alpha_1,\alpha_2}$ (see Figure 3): $\beta_{\min}(P) = \min_{Q \in A} \beta(P,Q)$.

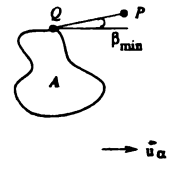


Figure 3: Definition of $\beta_{\min}$ with respect to an object A.

The spatial fuzzy set $\mu_\alpha(A)$ at point P is then defined as:

$$\mu_\alpha(A)(P) = f(\beta_{\min}(P)), \tag{24}$$

where f is a decreasing function of $[0, \pi]$ into $[0, 1]$. In our experiments, we have chosen a simple linear

function: $\mu_\alpha(A)(P) = \max(0, 1 - \frac{2\beta_{\min}(P)}{\pi})$, but other functions can be used, as trigonometric functions. We chose a function that sets the values of $\mu_\alpha(A)(P)$ to 0 as soon as $\beta_{\min}$ becomes greater than $\pi/2$. This avoids to get positive membership values for points having coordinates completely outside of the coordinate range of A in the desired direction.

In the fuzzy case, we propose a method which translates binary equations and propositions into fuzzy ones: in the binary case, we express that: $Q \in A$ and $f(\beta_{\min}) = \max_{Q \in A} f(\beta(P,Q))$ (since f is decreasing), which translates in fuzzy terms as:

$$\mu_\alpha(A)(P) = \max_{Q \in Supp(A)} t[\mu_A(Q), f(\beta(P,Q))], \quad (25)$$

where t is a t-norm.

Figure 4 illustrates a result obtained with this method, on one object of Figure 1. It shows that it well fits the intuition.

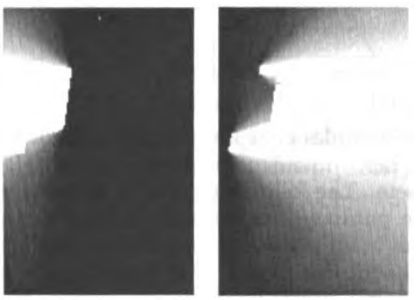


Figure 4: Left: fuzzy region representing the relationship "to the left of building 1" using Equation 25. Right: fuzzy region representing the relationship "to the right of building 1". Membership values vary from 0 (black) to 1 (white).

An advantage of this approach is its easy interpretation in terms of morphological operations. It can indeed be shown [Bloch(1999a)] that $\mu_\alpha(A)$ is exactly the fuzzy dilation of A by ν, where ν is the fuzzy structuring element defined on $\mathcal{S}$ as:

$$\forall P \in \mathcal{S},\ \nu(P) = f[\beta(O,P)], \quad (26)$$

with O as the center of the structuring element. For the linear function used in our experiments, the structuring element is:

$$\forall P \in \mathcal{S},\ \nu(P) = \max[0, 1 - \frac{2}{\pi} \arccos \frac{\vec{OP} \cdot \vec{u}_\alpha}{\|\vec{OP}\|}]. \quad (27)$$

It is represented in 2D in Figure 5.

Among the nice properties of this definition is invariance with respect to geometrical transformation

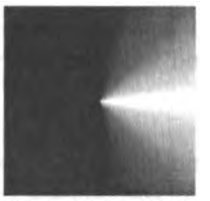


Figure 5: Structuring element ν for $\alpha = 0$.

(translation, rotation, scaling), which are requirements in object recognition. It also has a behavior that fits well the intuition if the distance to the reference object increases, and in case of concavities. These properties are detailed in [Bloch(1999a)], and several examples are shown. From an algorithmic point of view, an approximate method, based on propagation algorithms is proposed in [Bloch(1999a)], that reduces considerably the computation cost and the complexity.

6 EXAMPLE ON BRAIN STRUCTURES

In this Section we illustrate the knowledge representation method on a simple example on brain structures. This illustration comes from an atlas-based recognition method described in [Géraud et al.(1999)], that uses the type of knowledge representation formalized in this paper. A slice extracted from the atlas 3D volume is presented in Figure 6 (left); the right view shows the corresponding slice in a 3D magnetic resonance image (MRI) to be processed. The labeled image constitutes the iconic part of the model. The propositional part is constituted by expert knowledge about relationships between objects and expected radiometry of each structure. A spatial representation of this knowledge is derived, as presented in the previous sections.

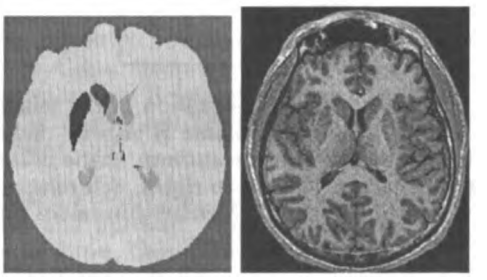


Figure 6: Slice extracted from a model atlas and from a MRI image. In the atlas, each grey level represents a different object we are interested in.

For instance, the recognition of a caudate nucleus in a

3D MRI image uses previous recognition of brain and lateral ventricles and following knowledge, illustrated in Figure 7:

- rough shape and localization are provided by the representation of the caudate nucleus in the atlas, and its fuzzy dilation to account for variability and for inexact matching between the model and the image,
- the caudate nucleus belongs to the brain (black) but is outside from both lateral ventricles (white components inside the brain),
- the caudate nucleus is lateral to the lateral ventricle.

These pieces of knowledge can be combined (also with information extracted from the image itself), which leads to a successful recognition of the caudate nucleus (see [Géraud et al.(1999)] for the fusion and recognition method).

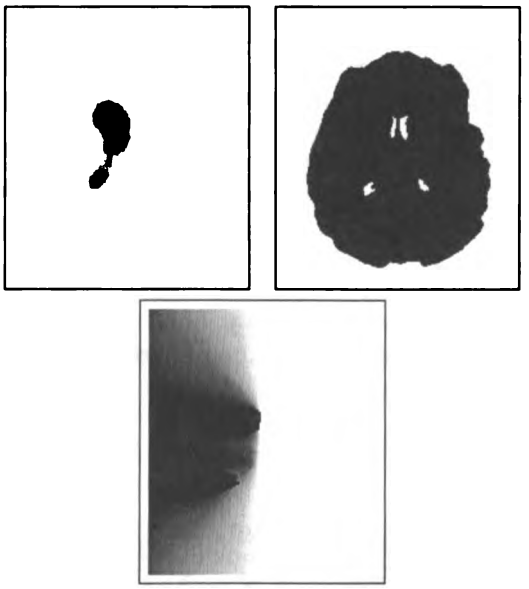


Figure 7: Information representation in the image space (only one slice of the 3D volume is shown), illustrating knowledge about one caudate nucleus: shape information (top left), set relationships (top right), and relative directional relationship (bottom). Membership values vary from 0 (white) to 1 (black).

Figure 8 illustrates the spatial representation of some knowledge about distances.

Figure 9 shows 3D views of some cerebral objects as defined in the atlas and as recognized in an MR image with our method [Géraud et al.(1999)]. They are correctly recognized although the size, the location and

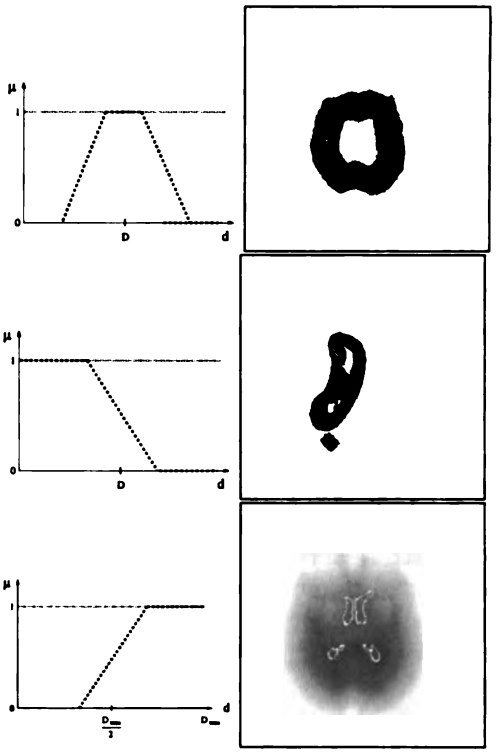


Figure 8: Examples of representation of knowledge about distances. Left: membership functions μ_n. Right: spatial fuzzy sets. The following types of knowledge are illustrated: the putamen has an approximately constant distance to the brain surface (top), the caudate nucleus is at a distance about less than D from the lateral ventricles (in white) (middle), lateral ventricles are inside the brain and at a distance larger than about D from the brain surface (bottom). The contours of the objects we are looking at are shown in white.

the morphology of these objects in the image significantly differ from their definitions in the atlas. Note in particular the good recognition of third and fourth ventricles, that are very difficult to segment directly from the image. Here the help of relationships to other structures is very important and conditions the quality of the results.

7 CONCLUSION

We proposed a method for representing knowledge about spatial relationships as fuzzy sets. As opposed to methods that aim at defining or assessing some relationships between objects, our aim here was to define space areas where relationships to one object are satisfied. The proposed approach uses fuzzy sets in order to account for imprecision, and mathematical morphology to derive most representations from set operations and to guarantee good properties, required for these representations. The obtained spatial fuzzy

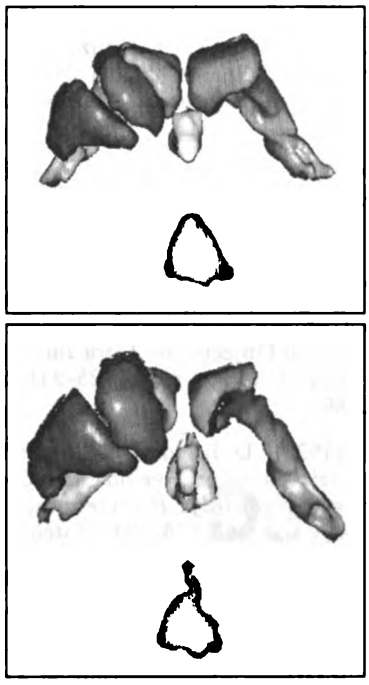


Figure 9: Recognition results. The upper view represents six objects from the model atlas: lateral ventricles (medium grey), third and fourth ventricles (light grey), caudate nucleus and putamen (dark grey). The lower view represents the equivalent objects recognized from a MRI acquisition. From [Géraud et al.(1999)].

sets can then be combined in order to guide search in the space, for spatial reasoning, and for recognition. We have shown a real example where representations of different relations are fused in order to focus on a region of a 3D image, and restrict this region until a brain structure is recognized, as the object that satisfies all these relations. Several structures are successively recognized. Another way to use the proposed representations is to check if some objects fit in the fuzzy areas corresponding to desired relations. The degree to which a point of the space satisfies a relation is directly given by the representation. But for sets of points, objects, or fuzzy objects, the adequation with the relation has to be computed, for instance by an aggregation method, or by a pattern matching approach. This allows to select objects that best satisfy a relation or several relations by combining either the representations before computing the adequation, or the degrees of adequation to each relation.

Acknowledgments: Thanks to Thierry Géraud for the example presented in Section 6, to Alessandro Saffiotti for enlightening discussions, to Henri Maître and to the reviewers for useful suggestions.

References

[Allen(1983)] J. Allen. Maintaining Knowledge about Temporal Intervals. *Comunications of the ACM*, 26 (11):832–843, 1983.

[Asher and Vieu(1995)] N. Asher and L. Vieu. Toward a Geometry of Common Sense: A Semantics and a Complete Axiomatization of Mereotopology. In *IJCAI'95*, pages 846–852, San Mateo, CA, 1995.

[Bloch(1999a)] I. Bloch. Fuzzy Relative Position between Objects in Image Processing: a Morphological Approach. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(7):657–664, 1999a.

[Bloch(1999b)] I. Bloch. On Fuzzy Distances and their Use in Image Processing under Imprecision. *Pattern Recognition*, 32(11):1873–1895, 1999b.

[Bloch and Maître(1995)] I. Bloch and H. Maître. Fuzzy Mathematical Morphologies: A Comparative Study. *Pattern Recognition*, 28(9):1341–1387, 1995.

[Bloch et al.(1997)] I. Bloch, H. Maître, and M. Anvari. Fuzzy Adjacency between Image Objects. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 5(6):615–653, 1997.

[Bloch et al.(1996)] I. Bloch, C. Pellot, F. Sureda, and A. Herment. Fuzzy Modelling and Fuzzy Mathematical Morphology applied to 3D Reconstruction of Blood Vessels by Multi-Modality Data Fusion. In D. Dubois R. Yager and H. Prade, editors, *Fuzzy Set Methods in Information Engineering: A Guided Tour of Applications*, chapter 5, pages 93–110. John Wiley & Sons, New-York, 1996.

[Borgefors(1996)] G. Borgefors. Distance Transforms in the Square Grid. In H. Maître, editor, *Progress in Picture Processing, Les Houches, Session LVIII, 1992*, chapter 1.4, pages 46–80. North-Holland, Amsterdam, 1996.

[Clementini and Felice(1997)] E. Clementini and O. Di Felice. Approximate Topological Relations. *International Journal of Approximate Reasoning*, 16:173–204, 1997.

[Cohn et al.(1997)] A. Cohn, B. Bennett, J. Gooday, and N. M. Gotts. Representing and Reasoning with Qualitative Spatial Relations about Regions. In O. Stock, editor, *Spatial and Temporal Reasoning*, pages 97–134. Kluwer, 1997.

[Dubois and Prade(1980)] D. Dubois and H. Prade. *Fuzzy Sets and Systems: Theory and Applications*. Academic Press, New-York, 1980.

[Dubois and Prade(1988)] D. Dubois and H. Prade. Weighted Fuzzy Pattern Matching. *Fuzzy Sets and Systems*, 28:313–331, 1988.

[Dutta(1991)] S. Dutta. Approximate Spatial Reasoning: Integrating Qualitative and Quantitative Constraints. *International Journal of Approximate Reasoning*, 5: 307–331, 1991.

[Freeman(1975)] J. Freeman. The Modelling of Spatial Relations. *Computer Graphics and Image Processing*, 4 (2):156–171, 1975.

[Gapp(1994)] K. P. Gapp. Basic Meanings of Spatial Relations: Computation and Evaluation in 3D Space. In *12th National Conference on Artificial Intelligence, AAAI-94*, pages 1393–1398, Seattle, Washington, 1994.

[Gasós and Ralescu(1997)] J. Gasós and A. Ralescu. Using Imprecise Environment Information for Guiding Scene Interpretation. *Fuzzy Sets and Systems*, 88:265–288, 1997.

[Gasós and Saffiotti(2000)] J. Gasós and A. Saffiotti. Using Fuzzy Sets to Represent Uncertain Spatial Knowledge. *Journal of Spatial Cognition and Computation*, 2000.

[Géraud et al.(1999)] T. Géraud, I. Bloch, and H. Maître. Atlas-guided Recognition of Cerebral Structures in MRI using Fusion of Fuzzy Structural Information. In *CIMAF'99 Symposium on Artificial Intelligence*, pages 99–106, La Havana, Cuba, 1999.

[Keller and Wang(1995)] J. M. Keller and X. Wang. Comparison of Spatial Relation Definitions in Computer Vision. In *ISUMA-NAFIPS'95*, pages 679–684, College Park, MD, 1995.

[Koczy(1988)] L. T. Koczy. On the Description of Relative Position of Fuzzy Patterns. *Pattern Recognition Letters*, 8:21–28, 1988.

[Krishnapuram et al.(1993)] R. Krishnapuram, J. M. Keller, and Y. Ma. Quantitative Analysis of Properties and Spatial Relations of Fuzzy Image Regions. *IEEE Transactions on Fuzzy Systems*, 1(3):222–233, 1993.

[Kuipers(1978)] B. Kuipers. Modeling Spatial Knowledge. *Cognitive Science*, 2:129–153, 1978.

[Kuipers and Levitt(1988)] B. J. Kuipers and T. S. Levitt. Navigation and Mapping in Large-Scale Space. *AI Magazine*, 9(2):25–43, 1988.

[Liu(1998)] J. Liu. A Method of Spatial Reasoning based on Qualitative Trigonometry. *Artificial Intelligence*, 98:137–168, 1998.

[Mangin et al.(1994)] J.-F. Mangin, I. Bloch, J. Lopez-Krahe, and V. Frouin. Chamfer Distances in Anisotropic 3D Images. In *EUSIPCO 94*, pages 975–978, Edinburgh, UK, 1994.

[Matsakis and Wendling(1999)] P. Matsakis and L. Wendling. A New Way to Represent the Relative Position between Areal Objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(7):634–642, 1999.

[Menger(1942)] K. Menger. Statistical Metrics. *Proc. National Academy of Siences USA*, 28:535–537, 1942.

[Miyajima and Ralescu(1994)] K. Miyajima and A. Ralescu. Spatial Organization in 2D Segmented Images: Representation and Recognition of Primitive Spatial Relations. *Fuzzy Sets and Systems*, 65:225–236, 1994.

[Peuquet(1988)] D. J. Peuquet. Representations of Geographical Space: Toward a Conceptual Synthesis. *Annals of the Association of American Geographers*, 78 (3):375–394, 1988.

[Pullar and Egenhofer(1988)] D. Pullar and M. Egenhofer. Toward Formal Definitions of Topological Relations Among Spatial Objects. In *Third Int.. Symposium on Spatial Data Handling*, pages 225–241, Sydney, Australia, 1988.

[Randell et al.(1992)] D. Randell, Z. Cui, and A. Cohn. A Spatial Logic based on Regions and Connection. In *Principles of Knowledge Representation and Reasoning KR'92*, pages 165–176, San Mateo, CA, 1992.

[Rosenfeld(1979)] A. Rosenfeld. Fuzzy Digital Topology. *Information and Control*, 40:76–87, 1979.

[Rosenfeld(1984)] A. Rosenfeld. The Fuzzy Geometry of Image Subsets. *Pattern Recognition Letters*, 2:311–317, 1984.

[Rosenfeld and Klette(1985)] A. Rosenfeld and R. Klette. Degree of Adjacency or Surroundness. *Pattern Recognition*, 18(2):169–177, 1985.

[Schweizer and Sklar(1963)] B. Schweizer and A. Sklar. Associative Functions and Abstract Semigroups. *Publ. Math. Debrecen*, 10:69–81, 1963.

[Udupa and Samarasekera(1996)] J. K. Udupa and S. Samarasekera. Fuzzy Connectedness and Object Definition: Theory, Algorithms, and Applications in Image Segmentation. *Graphical Models and Image Processing*, 58(3):246–261, 1996.

[Varzi(1996)] A. Varzi. Parts, Wholes, and Part-Whole Relations: The Prospects of Mereotopology. *Data and Knowledge Engineering*, 20(3):259–286, 1996.

[Vieu(1997)] L. Vieu. Spatial Representation and Reasoning in Artificial Intelligence. In O. Stock, editor, *Spatial and Temporal Reasoning*, pages 5–41. Kluwer, 1997.

[Wang and Keller(1999)] X. Wang and J. M. Keller. Human-based Spatial Relationship Generalization through Neural/Fuzzy Approaches. *Fuzzy Sets and Systems*, 101(1):5–20, 1999.

[Zadeh(1965)] L. A. Zadeh. Fuzzy Sets. *Information and Control*, 8:338–353, 1965.

[Zadeh(1975)] L. A. Zadeh. The Concept of a Linguistic Variable and its Application to Approximate Reasoning. *Information Sciences*, 8:199–249, 1975.

[Zwick et al.(1987)] R. Zwick, E. Carlstein, and D. V. Budescu. Measures of Similarity Among Fuzzy Concepts: A Comparative Analysis. *International Journal of Approximate Reasoning*, 1:221–242, 1987.

Description Logics

Matching in Description Logics with Existential Restrictions

Franz Baader and Ralf Küsters
LuFg Theoretical Computer Science, RWTH Aachen
Ahornstraße 55, 52074 Aachen, Germany
email: {baader,kuesters}@informatik.rwth-aachen.de

Abstract

Matching of concepts against patterns is a new inference task in Description Logics, which was originally motivated by applications of the CLASSIC system. Consequently, the work on this problem was until now mostly concerned with sublanguages of the CLASSIC language, which does not allow for existential restrictions.

This paper extends the existing work on matching in two directions. On the one hand, the question of what are the most "interesting" solutions of matching problems is explored in more detail. On the other hand, for languages with existential restrictions both, the complexity of deciding the solvability of matching problems and the complexity of actually computing sets of "interesting" matchers are determined. The results show that existential restrictions make these computational tasks more complex. Whereas for sublanguages of CLASSIC both problems could be solved in polynomial time, this is no longer possible for languages with existential restrictions.

1 Introduction

In description logics (DLs), the standard inference problems, like the subsumption and the instance problem, are now well-investigated. More recently, new types of inference problems have been introduced and investigated, like computing the least common subsumer of concepts (Cohen et al., 1992; Baader et al., 1999b) and matching concept descriptions against patterns.

This paper is concerned with the latter of these two inference problems, which has originally been introduced in (Borgida and McGuinness, 1996; McGuinness, 1996) to help filter out the unimportant aspects of large concepts appearing in knowledge bases of the CLASSIC DL system (Brachman et al., 1991; Borgida et al., 1989). More recently, matching (as well as the more general problem of unification) has been proposed as a tool for detecting redundancies in knowledge bases (Baader and Narendran, 1998) and to support the integration of knowledge bases by prompting possible interschema assertions to the integrator (Borgida and Küsters, 1999).

All three applications have in common that one wants to search the knowledge base for concepts having a certain (not completely specified) form. This "form" can be expressed with the help of so-called *concept patterns*, i.e., concept descriptions containing variables (which stand for descriptions). For example, assume that we want to find concepts that are concerned with individuals having a son and a daughter sharing some characteristic. This can be expressed by the pattern $D := \exists \mathsf{has\text{-}child}.(\mathsf{Male} \sqcap X) \sqcap \exists \mathsf{has\text{-}child}.(\mathsf{Female} \sqcap X)$, where X is a variable standing for the common characteristic. The concept description $C := \exists \mathsf{has\text{-}child}.(\mathsf{Tall} \sqcap \mathsf{Male}) \sqcap \exists \mathsf{has\text{-}child}.(\mathsf{Tall} \sqcap \mathsf{Female})$ matches this pattern in the sense that, if we replace the variable X by the description Tall, the pattern becomes *equivalent* to the description. Thus, the substitution $\sigma := \{X \mapsto \mathsf{Tall}\}$ is a *matcher modulo equivalence* of the matching problem $C \equiv^? D$. Note that not only the fact that there is a matcher is of interest, but also the matcher itself, since it tells us what is the common characteristic of the son and the daughter.

Looking for such an exact match (called *matching modulo equivalence* in the following) is not always appropriate, though. In our example, using matching modulo equivalence means that all the additional characteristics of the son and daughter mentioned in the

concept must be common to both. Thus, the description $C' := \exists$has-child.(Tall $\sqcap$ Male $\sqcap$ Talkative) $\sqcap$ $\exists$has-child.(Tall $\sqcap$ Female $\sqcap$ Quiet) does not match the pattern modulo equivalence. *Matching modulo subsumption* only requires that, after the replacement, the pattern subsumes the description. Thus, the substitution σ from above is a *matcher modulo subsumption* of the matching problem $C' \sqsubseteq^? D$.

Previous results on matching in DLs were mostly concerned with sublanguages of the CLASSIC description language, which does not allow for existential restrictions of the kind used in our example. A polynomial-time algorithm for computing matchers modulo subsumption for a rather expressive DL was introduced in (Borgida and McGuinness, 1996). The main drawback of this algorithm is that it requires the concept patterns to be in structural normal form, and thus it cannot handle arbitrary matching problems. In addition, the algorithm is incomplete, i.e., it does not always find a matcher, even if one exists. For the DL $\mathcal{ACN}$, a polynomial-time algorithm for matching modulo subsumption and equivalence was presented in (Baader et al., 1999a). This algorithm is complete and it applies to arbitrary patterns.

Motivated by an application in chemical process engineering (Baader and Sattler, 1996), which requires existential restrictions, the main purpose of this paper is to investigate matching in DLs allowing for existential restrictions. We will show that existential restrictions make matching more complex in two respects. First, whereas matching in the DLs considered in (Borgida and McGuinness, 1996; Baader et al., 1999a) is polynomial, even deciding the existence of matchers is an NP-complete problem in the presence of existential restrictions (Section 3). Second, the algorithms described in (Borgida and McGuinness, 1996; Baader et al., 1999a) always compute the least matcher (w.r.t. subsumption of substitutions; see the definition of $\sqsubseteq_s$ in Section 4) of the given matching problem. For languages with existential restrictions, such a unique least matcher need not exist. However, the set of minimal matchers is finite (though possibly exponential in the size of the matching problem), and we will show how to compute this set (Section 5). It has turned out, however, that the minimal matchers are not necessarily the most interesting ones since they may contain certain redundancies. Thus, one also needs a kind of post-processing step that removes these redundancies (Section 6). Since giving an answer to the question of what are good sets of matchers is a crucial and non-trivial problem, which has not been explored satisfactorily so far, we will treat this question in a separate section (Section 4). A more detailed presentation and complete proofs of all the results stated in this paper can be found in (Baader and Küsters, 1999).

Syntax	Semantics
$\top$	Δ
$C \sqcap D$	$C^I \cap D^I$
$\exists r.C$	$\{x \in \Delta \mid \exists y : (x,y) \in r^I \wedge y \in C^I\}$
$\forall r.C$	$\{x \in \Delta \mid \forall y : (x,y) \in r^I \rightarrow y \in C^I\}$
$\neg P$, $P \in N_C$	$\Delta \setminus P^I$
$\bot$	$\emptyset$

Table 1: Syntax and semantics of concept descriptions.

2 Preliminaries

Concept descriptions are inductively defined with the help of a set of concept *constructors*, starting with a set N_C of *concept names* and a set N_R of *role names*. In this paper, we consider concept descriptions built from the constructors shown in Table 1. In the description logic $\mathcal{EL}$, concept descriptions are formed using the constructors top-concept ($\top$), conjunction ($C \sqcap D$), and existential restriction ($\exists r.C$). The description logic $\mathcal{ALE}$ additionally provides us with value restrictions ($\forall r.C$), primitive negation ($\neg P$), and the bottom-concept ($\bot$).

As usual, the semantics of concept descriptions is defined in terms of an *interpretation* $\mathcal{I} = (\Delta, \cdot^I)$. The domain Δ of $\mathcal{I}$ is a non-empty set and the interpretation function $\cdot^I$ maps each concept name $P \in N_C$ to a set $P^I \subseteq \Delta$ and each role name $r \in N_R$ to a binary relation $r^I \subseteq \Delta \times \Delta$. The extension of $\cdot^I$ to arbitrary concept descriptions is defined inductively, as shown in the second column of Table 1.

One of the most important traditional inference services provided by DL systems is computing the subsumption hierarchy. The concept description C is *subsumed* by the description D ($C \sqsubseteq D$) iff $C^I \subseteq D^I$ holds for all interpretations $\mathcal{I}$. The concept descriptions C and D are *equivalent* ($C \equiv D$) iff they subsume each other.

In order to define concept patterns, we additionally need a set $\mathcal{X}$ of concept variables, which we assume to be disjoint from $N_C \cup N_R$. Informally, an $\mathcal{ALE}$-concept pattern is an $\mathcal{ALE}$-concept description over the concept names $N_C \cup \mathcal{X}$ and the role names N_R, with the only exception that primitive negation must not be applied to variables.

Definition 1 *The set of all $\mathcal{ALE}$-concept patterns over N_C, N_R, and $\mathcal{X}$ is inductively defined as follows:*

- *Every concept variable $X \in \mathcal{X}$ is a pattern.*
- *Every $\mathcal{ALE}$-concept description over N_C and N_R is a pattern.*
- *If D_1 and D_2 are concept patterns, then $D_1 \sqcap D_2$ is a concept pattern.*
- *If D is a concept pattern and r is a role name, then $\forall r.D$ and $\exists r.D$ are concept patterns.*

The notion of a pattern (and also the notions "substitution" and "matching problem" introduced below) can be restricted to $\mathcal{EL}$ in the obvious way. For example, if X, Y are concept variables, r a role name, and A, B concept names, then $D := A \sqcap X \sqcap \exists r.(B \sqcap Y)$ is both an $\mathcal{EL}$- and $\mathcal{ALE}$-concept pattern, but $\neg X$ is neither an $\mathcal{ALE}$- nor an $\mathcal{EL}$-concept pattern.

A *substitution* σ is a mapping from $\mathcal{X}$ into the set of all $\mathcal{ALE}$-concept descriptions. This mapping is extended to concept patterns in the usual way, i.e.,

- $\sigma(C) := C$ for all $\mathcal{ALE}$-concept descriptions C,
- $\sigma(D_1 \sqcap D_2) := \sigma(D_1) \sqcap \sigma(D_2)$,
- $\sigma(\forall r.D) := \forall r.\sigma(D)$ and $\sigma(\exists r.D) := \exists r.\sigma(D)$.

For example, if we apply the substitution $\sigma := \{X \mapsto A \sqcap B,\ Y \mapsto A\}$ to the pattern D from above, we obtain the description $\sigma(D) = A \sqcap A \sqcap B \sqcap \exists r.(B \sqcap A)$. The result of applying a substitution to an $\mathcal{ALE}$-concept pattern is always an $\mathcal{ALE}$-concept description. Note that this would no longer be the case if negation were allowed in front of concept variables. In fact, $\sigma(\neg X) = \neg(A \sqcap B)$ cannot be expressed in $\mathcal{ALE}$.

Definition 2 *Let C be an $\mathcal{ALE}$-concept description and D an $\mathcal{ALE}$-concept pattern. Then, $C \equiv^? D$ is an* $\mathcal{ALE}$-matching problem modulo equivalence *and $C \sqsubseteq^? D$ is an* $\mathcal{ALE}$-matching problem modulo subsumption. *The substitution σ is a* matcher *of $C \equiv^? D$ iff $C \equiv \sigma(D)$, and it is a* matcher *of $C \sqsubseteq^? D$ iff $C \sqsubseteq \sigma(D)$.*

3 Complexity of the decision problem

In this section, we are interested in the complexity of deciding whether a given matching problem has a matcher or not. Our results are summarized in Table 2. The first and the second row of the table refer to matching modulo subsumption and matching modulo equivalence, respectively.

	$\mathcal{EL}$	$\mathcal{ALE}$
subsumption	P	NP-complete
equivalence	NP-complete	NP-complete

Table 2: Deciding solvability of matching problems

First, note that patterns are not required to contain variables. Consequently, matching modulo subsumption (equivalence) is at least as hard as subsumption (equivalence). Thus, NP-completeness of subsumption in $\mathcal{ALE}$ (Donini et al., 1992) yields the hardness part of the second column of Table 2. Second, for the languages $\mathcal{ALE}$ and $\mathcal{EL}$, matching modulo subsumption can be reduced to subsumption: $C \sqsubseteq^? D$ has a matcher iff $\sigma_\top := \{X \mapsto \top \mid X \in \mathcal{X}\}$ is a matcher of $C \sqsubseteq^? D$. Thus, the known complexity results for subsumption in $\mathcal{ALE}$ and $\mathcal{EL}$ (Donini et al., 1992; Baader et al., 1999b) complete the first row of Table 2. Third, NP-hardness of matching modulo equivalence for $\mathcal{EL}$ can be shown by a reduction of SAT. The reduction in (Baader and Küsters, 1999) uses only a fixed number of role names. Here, we give a simpler reduction for which, however, the number of role names grows with the formula.

Lemma 3 *Deciding the solvability of matching problems modulo equivalence in $\mathcal{EL}$ is NP-hard.*

PROOF. Let $\phi = p_1 \wedge \cdots \wedge p_m$ be a propositional formula in conjunctive normal form and let $\{x_1, \ldots, x_n\}$ be the propositional variables of this problem. For these variables, we introduce the concept variables $\{X_1, \ldots, X_n, \overline{X}_1, \ldots, \overline{X}_n\}$. Furthermore, we need concept names A and B as well as role names $r_1, \ldots, r_n$ and $s_1, \ldots, s_m$.

First, we specify a matching problem $C_n \equiv^? D_n$ that encodes the truth values of the n propositional variables:

$$\begin{aligned} C_n &:= \exists r_1.A \sqcap \exists r_1.B \sqcap \cdots \sqcap \exists r_n.A \sqcap \exists r_n.B, \\ D_n &:= \exists r_1.X_1 \sqcap \exists r_1.\overline{X}_1 \sqcap \cdots \sqcap \exists r_n.X_n \sqcap \exists r_n.\overline{X}_n. \end{aligned}$$

The matchers of this problem are exactly the substitutions that replace X_i by A and $\overline{X}_i$ by B (corresponding to $x_i = \mathit{true}$), or vice versa (corresponding to $x_i = \mathit{false}$).

In order to encode ϕ, we introduce a concept pattern D_{p_i} for each conjunct p_i. For example, if $p_i = x_1 \vee \overline{x}_2 \vee x_3 \vee \overline{x}_4$, then $D_{p_i} := X_1 \sqcap \overline{X}_2 \sqcap X_3 \sqcap \overline{X}_4 \sqcap B$. The whole formula is then represented by the matching problem $C_\phi \equiv^? D_\phi$, where

$$\begin{aligned} C_\phi &:= \exists s_1.(A \sqcap B) \sqcap \cdots \sqcap \exists s_m.(A \sqcap B), \\ D_\phi &:= \exists s_1.D_{p_1} \sqcap \cdots \sqcap \exists s_m.D_{p_m}. \end{aligned}$$

This matching problem ensures that, among all the variables in D_{p_i}, at least one must be replaced by A. This corresponds to the fact that, within one conjunct p_i, there must be at least one literal that evaluates to *true*. Note that we need the concept B in D_{p_i} to cover the case where all variables in D_{p_i} are substituted with A.

We combine the two matching problems introduced above into a single problem $C_n \sqcap C_\phi \equiv^? D_n \sqcap D_\phi$. It is easy to verify that ϕ is satisfiable iff this matching problem is solvable. ∎

It remains to show that matching modulo equivalence in $\mathcal{EL}$ and $\mathcal{ALE}$ can in fact be decided in nondeterministic polynomial time. This is an easy consequence of the following lemma.

Lemma 4 *If an $\mathcal{EL}$- or $\mathcal{ALE}$-matching problem modulo equivalence has a matcher, then it has one of size polynomially bounded by the size of the problem. Furthermore, this matcher uses only concept and role names already contained in the matching problem.*

The lemma (together with the known complexity results for subsumption) shows that the following can be realized in NP: "guess" a substitution satisfying the given size bound, and then test whether it is a matcher.

For $\mathcal{EL}$, the polynomial bound stated in the lemma can be derived from our algorithm for computing so-called s-co-complete sets of matchers presented in Section 6. At the end of Section 6, we will also comment on the (quite involved) proof of Lemma 4 for $\mathcal{ALE}$.

4 Solutions of matching problems

In this section and in Section 5.1, we will use the $\mathcal{EL}$-concept description C^1_{ex} and the pattern D^1_{ex} shown in Figure 1 as our running example. (The graphical representation will be explained later on.)

It is easy to see that the substitution $\sigma_\top$ is a matcher of $C^1_{\text{ex}} \sqsubseteq^? D^1_{\text{ex}}$, and thus this matching problem modulo subsumption is indeed solvable. However, the matcher $\sigma_\top$ is obviously not an interesting one. We are interested in matchers that bring us as close as possible to the description C^1_{ex}. In this sense, the matcher $\sigma_1 := \{X \mapsto \mathsf{W} \sqcap \exists \mathsf{hc}.\mathsf{W}, Y \mapsto \mathsf{W}\}$ is better than $\sigma_\top$, but still not optimal. In fact, $\sigma_2 := \{X \mapsto \mathsf{W} \sqcap \exists \mathsf{hc}.\mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P}), Y \mapsto \mathsf{W} \sqcap \mathsf{D}\}$ is better than σ_1 since it satisfies $C^1_{\text{ex}} \equiv \sigma_2(D^1_{\text{ex}}) \sqsubset \sigma_1(D^1_{\text{ex}})$.

We formalize this intuition with the help of the following precedence ordering on matchers. For a given matching problem $C \sqsubseteq^? D$ and two matchers σ, τ we define

$$\sigma \sqsubseteq_i \tau \text{ iff } \sigma(D) \sqsubseteq \tau(D).$$

Here "i" stands for "instance". Two matchers σ, τ are *i-equivalent* ($\sigma \equiv_i \tau$) iff $\sigma \sqsubseteq_i \tau$ and $\tau \sqsubseteq_i \sigma$. A matcher σ is called *i-minimal* iff, for every matcher τ, $\tau \sqsubseteq_i \sigma$ implies $\tau \equiv_i \sigma$. We are interested in *computing i-minimal matchers*; more precisely, we want to obtain at least one i-minimal matcher for each of the minimal i-equivalence classes (i.e., i-equivalence classes of i-minimal matchers). Since an i-equivalence class usually contains more than one matcher, the question is which ones to prefer.

In (Baader et al., 1999a), it is shown that a given $\mathcal{ALN}$-matching problem always has a unique minimal i-equivalence class, and that this class is the class of the least matcher w.r.t. the ordering

$$\sigma \sqsubseteq_s \tau \text{ iff } \sigma(X) \sqsubseteq \tau(X) \text{ for all } X \in \mathcal{X},$$

where "s" stands for "substitution". The matcher σ is a *least matcher w.r.t.* $\sqsubseteq_s$ iff $\sigma \sqsubseteq_s \tau$ for all matchers τ. The notions s-minimal, s-equivalent, etc. are defined in the obvious way.

For $\mathcal{EL}$ and $\mathcal{ALE}$, things are quite different. As illustrated by the example $\exists r.A \sqcap \exists r.B \sqsubseteq^? \exists r.X$, a given matching problem may have several non-equivalent i-minimal (s-minimal) matchers: the substitutions $\{X \mapsto A\}$ and $\{X \mapsto B\}$ are both i- and s-minimal, and they are obviously neither i- nor s-equivalent. It can be shown (Baader and Küsters, 1999) that the set of all s-minimal matchers (up to s-equivalence) also contains all i-minimal matchers (up to i-equivalence). However, the s-minimal matchers are usually not the best representatives of their i-equivalence class.

In our running example, σ_2 is a least and therefore i-minimal matcher. Nevertheless, it is not the one we really want to compute since it contains redundancies, i.e., expressions that are not really necessary for obtaining the instance $\sigma_2(D^1_{\text{ex}})$ (modulo equivalence). In fact, σ_2 contains two different kinds of redundancies. First, the existential restriction $\exists \mathsf{hc}.\mathsf{W}$ in $\sigma_2(X)$ is redundant since removing it still yields a concept description equivalent to $\sigma_2(X)$. Second, W in $\sigma_2(Y)$ is redundant in that the substitution obtained by deleting W from $\sigma_2(Y)$ still yields the same instance of D^1_{ex} (although the substitution obtained this way is no longer s-equivalent to σ_2). In our example, the only i-minimal matcher (modulo associativity and commutativity of concept conjunction) that is free of redundancies in this sense is $\sigma_3 := \{X \mapsto \mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P}), Y \mapsto \mathsf{D}\}$.

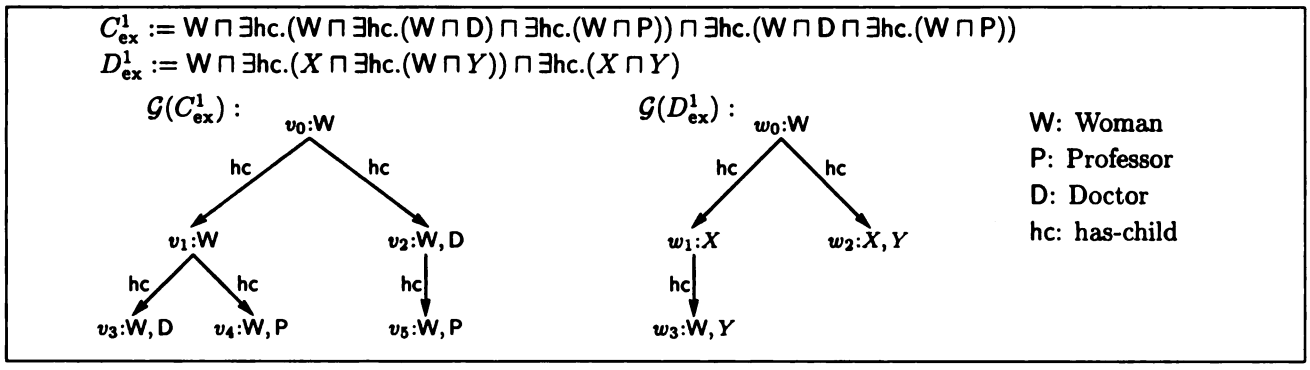


Figure 1: $\mathcal{EL}$-concept description and pattern, and their $\mathcal{EL}$-description trees.

We want to *compute i-minimal matchers that are reduced*, i.e., free of redundancies. It remains to formalize the notion "reduced" more rigorously. For this purpose, we need the notion of a subdescription.

Definition 5 *For an $\mathcal{ALE}$-concept description C, the $\mathcal{ALE}$-concept description $\widehat{C}$ is a* subdescription *of C ($\widehat{C} \preceq_d C$) iff* (i) $\widehat{C} = C$; *or* (ii) $\widehat{C} = \bot$; *or* (iii) *$\widehat{C}$ is obtained from C by removing some (negated) concept names, value restrictions, or existential restrictions on the top-level of C, and for all remaining value/existential restrictions $\forall r.E/\exists r.E$ replacing E by a subdescription of E.*

Note that, if everything is removed from C, then $\widehat{C} = \top$. For $\mathcal{EL}$, subdescriptions are defined analogously; clearly, (ii) must be removed for $\mathcal{EL}$. The concept description $\widehat{C}$ is a *strict subdescription* of C iff $\widehat{C} \preceq_d C$ and $\widehat{C} \neq C$.

Definition 6 *The $\mathcal{ALE}$-concept description C is called* reduced *iff there does not exist a strict subdescription of C that is equivalent to C.*

For example, $\sigma_3(X)$ is reduced, whereas $\sigma_2(X)$ is not. Reduced concept descriptions are unique, provided that they are also in $\forall$-normal form. An $\mathcal{ALE}$-concept description C is in *$\forall$-normal form* iff the $\forall$-rule $\forall r.E \sqcap \forall r.F \longrightarrow \forall r.(E \sqcap F)$ cannot be applied to C. Clearly, every concept description can easily be transformed (in polynomial time) into its $\forall$-normal form by exhaustively applying the $\forall$-rule.

Lemma 7 *(Baader and Küsters, 1999) Equivalent and reduced $\mathcal{ALE}$-concept descriptions in $\forall$-normal form are equal up to associativity and commutativity of concept conjunction.*

The ordering $\preceq_d$ can be extended to substitutions in the obvious way:

$$\sigma \preceq_d \tau \text{ iff } \sigma(X) \preceq_d \tau(X) \text{ for all } X \in \mathcal{X}.$$

The matcher σ of $C \sqsubseteq^? D$ is called *reduced* iff it is a d-minimal matcher (i.e., minimal w.r.t. $\preceq_d$). Note that, given a reduced matcher σ, every concept description $\sigma(X)$ is reduced. However, as illustrated by our running example (removal of W in $\sigma_2(Y)$), just replacing the descriptions $\sigma(X)$ by equivalent reduced descriptions does not necessarily yield a reduced matcher.

To sum up, given a matching problem $C \sqsubseteq^? D$, we want to compute matchers that are i-minimal and reduced. It should be noted that a given i-equivalence class of matchers may contain different reduced matchers. Since reduced and s-equivalent matchers are equal up to associativity and commutativity of conjunction, it is, however, sufficient to compute the reduced matchers up to s-equivalence.

Our approach for computing i-minimal and reduced matchers of $C \sqsubseteq^? D$ proceeds in two steps (which we consider in more detail in the next two sections):

1. Compute the set of all i-minimal matchers of $C \sqsubseteq^? D$ up to i-equivalence (i.e., one matcher for each i-equivalence class).
2. For each i-minimal matcher σ computed in the first step, compute the d-minimal matchers up to s-equivalence of $\sigma(D) \equiv^? D$.

Of course, if we are interested in matching modulo equivalence in the first place, we just apply the second step to $C \equiv^? D$.

5 Computing i-minimal matchers

In this section, we show how to compute the set of all i-minimal matchers up to i-equivalence for a given matching problem $C \sqsubseteq^? D$. In fact, the algorithms for $\mathcal{EL}$ and $\mathcal{ALE}$ that we will present below solve a slightly different problem: they compute so-called s-complete sets of matchers.

Definition 8 *A set of matchers is called* s-complete *iff it contains (at least) all s-minimal matchers up to s-equivalence. It is called* minimal s-complete, *if it consists of one representative of every s-equivalence class of the s-minimal matchers.*

Analogously, one can define (minimal) i-complete sets. A simple proof shows the following relationship between s- and i-complete sets.

Lemma 9 *Every s-complete set is also i-complete.*

Given an s-complete set, one can therefore in a post-processing step use the subsumption algorithm (for $\mathcal{EL}$ or $\mathcal{ALE}$) to determine a minimal i-complete set.

Mainly for didactic reasons, we present the algorithm for computing s-complete sets of matchers both for $\mathcal{EL}$ and $\mathcal{ALE}$. In general, an algorithm for a given DL does not necessarily work for its sublanguages since the set of potential matchers changes. In this particular case, however, the algorithm for $\mathcal{ALE}$ applied to $\mathcal{EL}$-matching problems only yields matchers in $\mathcal{EL}$.

5.1 Computing s-complete sets in $\mathcal{EL}$

The algorithm for computing s-complete sets of matchers in $\mathcal{EL}$ is based on a characterization of subsumption between $\mathcal{EL}$-concepts via homomorphisms between the corresponding description trees. This characterization has been introduced in (Baader et al., 1999b) for the purpose of computing the least common subsumer (lcs) of $\mathcal{EL}$-concepts. Before introducing the matching algorithm, we briefly recall the characterization of subsumption.

Characterizing subsumption in $\mathcal{EL}$

Definition 10 *An* $\mathcal{EL}$-description tree *is a tree of the form* $\mathcal{G} = (V, E, v_0, \ell)$ *where*

- V *is a finite set of* nodes;
- $E \subseteq V \times N_R \times V$ *is a finite set of* edges *labeled with role names* r *($\exists$-edges);*
- $v_0 \in V$ *is the* root *of* $\mathcal{G}$;
- ℓ *is a* labeling function *mapping the nodes in* V *to finite subsets of* N_C. *The empty label corresponds to the top-concept.*

The *$\mathcal{EL}$-description tree* $\mathcal{G}(C)$ corresponding to the $\mathcal{EL}$-concept description C simply reflects the syntactic structure of the description. For example, the description tree corresponding to the $\mathcal{EL}$-concept description $C^1_{\mathbf{ex}}$ of our example is depicted on the left-hand side of Figure 1. For an $\mathcal{EL}$-concept description C and a node v in the corresponding description tree $\mathcal{G}(C)$, we denote the part of C corresponding to v by C_v. In our example, we have, for instance, $C^1_{\mathbf{ex},v_1} = \mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{D}) \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P})$.

Definition 11 *A mapping* $\varphi : V_H \longrightarrow V_G$ *from an* $\mathcal{EL}$*-description tree* $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$ *to an* $\mathcal{EL}$*-description tree* $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ *is called* homomorphism *iff the following conditions are satisfied:*

1. $\varphi(w_0) = v_0$,
2. *for all* $v \in V_H$ *we have* $\ell_H(v) \subseteq \ell_G(\varphi(v))$,
3. *for all* $vrw \in E_H$, *we have* $\varphi(v)r\varphi(w) \in E_G$.

Now, subsumption can be characterized as follows:

Lemma 12 *(Baader et al., 1999b) Given two* $\mathcal{EL}$*-concept descriptions* C, D, *we have* $C \sqsubseteq D$ *iff there exists a homomorphism from* $\mathcal{G}(D)$ *into* $\mathcal{G}(C)$.

The $\mathcal{EL}$-matching algorithm

In order to employ this lemma for deriving the $\mathcal{EL}$-matching algorithm, we need to generalize the notions introduced above to concept patterns. $\mathcal{EL}$-description trees can be extended to concept patterns by simply treating variables like concept names. For example, the concept pattern $D^1_{\mathbf{ex}}$ in the example yields the description tree depicted on the right-hand side of Figure 1. When *extending* the notion of a *homomorphism to description trees representing concept patterns*, we simply ignore the concept variables, i.e., the second condition must hold only for non-variable concept names.

In our example, there are six homomorphisms from $\mathcal{G}(D^1_{\mathbf{ex}})$ into $\mathcal{G}(C^1_{\mathbf{ex}})$. We consider the ones mapping w_i onto v_i for $i = 0, 1, 2$ and w_3 onto v_3 or w_3 onto v_4, which we denote by φ_1 and φ_2, respectively.

Input: $\mathcal{EL}$-matching problem $C \sqsubseteq^? D$.
Output: s-complete set $\mathcal{C}$ for $C \sqsubseteq^? D$.

$\mathcal{C} := \emptyset$;
For all homomorphisms φ from
  $\mathcal{G}(D) = (V, E, v_0, \ell)$ into $\mathcal{G}(C)$ do
    Define τ by $\tau(X) := lcs\{C_{\varphi(v)} \mid X \in \ell(v)\}$
      for all variables X in D;
    $\mathcal{C} := \mathcal{C} \cup \{\tau\}$;

Figure 2: The $\mathcal{EL}$-matching algorithm

The matching algorithm described in Figure 2 constructs substitutions τ such that $C \sqsubseteq \tau(D)$, i.e., there is a homomorphism from $\mathcal{G}(\tau(D))$ into $\mathcal{G}(C)$. This is achieved by first computing all homomorphisms from $\mathcal{G}(D)$ into $\mathcal{G}(C)$. The remaining problem is that a variable X may occur more than once in D. Thus, we cannot simply define $\tau(X)$ as $C_{\varphi(v)}$ where v is such that X occurs in the label of v. Since there may exist several nodes v with this property, we take the least common subsumer of the corresponding parts of C. The reason for taking the *least* common subsumer is that we want to compute substitutions that are as small as possible w.r.t. $\sqsubseteq_s$. Recall that E is the *least common subsumer* (lcs) of $E_1, \ldots, E_n$ ($lcs(E_1, \ldots, E_n)$ for short) iff (i) E subsumes $E_1, \ldots, E_n$ and (ii) E is the least concept description w.r.t. subsumption that satisfies (i), i.e., for every concept description E', if $E' \sqsupseteq E_1, \ldots, E_n$, then $E' \sqsupseteq E$. Algorithms for computing the lcs of $\mathcal{EL}$- and $\mathcal{ALE}$-concept descriptions have been described in (Baader et al., 1999b).

In our example, the homomorphism φ_1 yields the substitution τ_1:

$$\begin{aligned} \tau_1(X) &:= lcs\{C^1_{\mathsf{ex},v_1},\ C^1_{\mathsf{ex},v_2}\} \equiv \mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P}), \\ \tau_1(Y) &:= lcs\{C^1_{\mathsf{ex},v_2},\ C^1_{\mathsf{ex},v_3}\} \equiv \mathsf{W} \sqcap \mathsf{D}, \end{aligned}$$

whereas φ_2 yields the substitution τ_2:

$$\begin{aligned} \tau_2(X) &:= lcs\{C^1_{\mathsf{ex},v_1},\ C^1_{\mathsf{ex},v_2}\} \equiv \mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P}), \\ \tau_2(Y) &:= lcs\{C^1_{\mathsf{ex},v_2},\ C^1_{\mathsf{ex},v_4}\} \equiv \mathsf{W}. \end{aligned}$$

The substitution τ_1 is an i-minimal matcher, but τ_2 is neither i-minimal nor s-minimal. Therefore, τ_2 will be removed in the post-processing step when extracting a *minimal* i-complete set from the computed s-complete one. By applying Lemma 12, it is easy to show:

Theorem 13 *(Baader and Küsters, 1999) The algorithm described in Figure 2 always computes an s-complete set of matchers for a given $\mathcal{EL}$-matching problem modulo subsumption.*

5.2 Computing s-complete sets in $\mathcal{ALE}$

The algorithm for computing s-complete sets for $\mathcal{ALE}$-matching problems modulo subsumption is similar to the one for $\mathcal{EL}$. However, due to inconsistent concepts expressible in $\mathcal{ALE}$ and the interaction between existential and value restrictions, things become more complicated.

As before, the algorithm is based on the characterization of subsumption via homomorphisms between description trees, which we briefly recall in the following (see (Baader et al., 1999b) for more details).

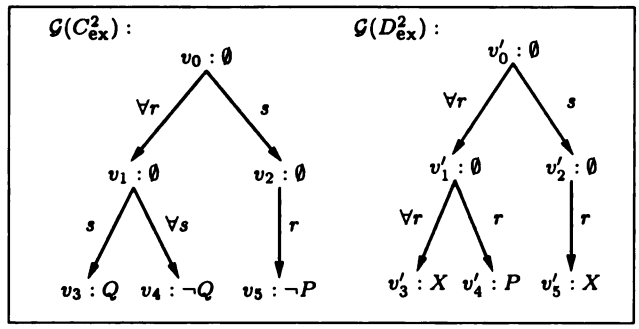


Figure 3: $\mathcal{ALE}$-description trees.

Characterizing subsumption in $\mathcal{ALE}$

The notion of $\mathcal{EL}$-description trees is generalized to $\mathcal{ALE}$ in a straightforward manner: (i) In addition to $\exists$-edges labeled with role names r, $\mathcal{ALE}$-description trees may also contain $\forall$-edges labeled $\forall r$; (ii) beside concept names and variables, labels of nodes in $\mathcal{ALE}$-description trees may also contain negated concept names as well as the bottom-concept.

Again, any $\mathcal{ALE}$-concept description/pattern C can be translated into a corresponding $\mathcal{ALE}$-description tree $\mathcal{G}(C)$. For example, the description trees $\mathcal{G}(C^2_{\mathsf{ex}})$ and $\mathcal{G}(D^2_{\mathsf{ex}})$ corresponding to the $\mathcal{ALE}$-concept description C^2_{ex} and the concept pattern D^2_{ex} defined as

$$\begin{aligned} C^2_{\mathsf{ex}} &:= \forall r.(\exists s.Q \sqcap \forall s.\neg Q) \sqcap \exists s.\exists r.\neg P, \\ D^2_{\mathsf{ex}} &:= \forall r.(\forall r.X \sqcap \exists r.P) \sqcap \exists s.\exists r.X \end{aligned}$$

are depicted in Figure 3.

In order to extend the characterization of subsumption from $\mathcal{EL}$ to $\mathcal{ALE}$, the notion of a homomorphism must be extended and concept descriptions need to be normalized before turning them into description trees.

Obviously, a homomorphism between $\mathcal{ALE}$-description trees must distinguish between $\forall$- and $\exists$-edges. More important, a homomorphism must be allowed to map a node and all its successors onto an inconsistent node, i.e., a node whose label contains $\bot$.

Definition 14 *A mapping $\varphi : V_H \longrightarrow V_G$ from an $\mathcal{ALE}$-description tree $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$ to an $\mathcal{ALE}$-description tree $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ is called* homomorphism *iff the following conditions are satisfied:*

1. $\varphi(w_0) = v_0$,
2. *for all $v \in V_H$ we have $\ell_H(v) \subseteq \ell_G(\varphi(v))$ or $\bot \in \ell_G(\varphi(v))$,*
3. *for all $vrw \in E_H$, either $\varphi(v)r\varphi(w) \in E_G$, or $\varphi(v) = \varphi(w)$ and $\bot \in \ell_G(\varphi(v))$, and*

4. *for all* $v\forall r w \in E_H$, *either* $\varphi(v)\forall r\varphi(w) \in E_G$, *or* $\varphi(v) = \varphi(w)$ *and* $\bot \in \ell_G(\varphi(v))$.

The main purpose of the normalization rules on $\mathcal{ALE}$-concept descriptions (see below) is to make implicitly inconsistent parts of C explicit, and to propagate value restrictions onto existential restrictions and other value restrictions.

Definition 15 *Let* E, F *be two* $\mathcal{ALE}$*-concept descriptions,* $r \in N_R$, *and* $P \in N_C$. *The* $\mathcal{ALE}$-normalization rules *are defined as follows*

$$\begin{aligned} P \sqcap \neg P,\ \exists r.\bot,\ E \sqcap \bot &\longrightarrow \bot, \\ \forall r.E \sqcap \exists r.F &\longrightarrow \forall r.E \sqcap \exists r.(E \sqcap F), \\ \forall r.E \sqcap \forall r.F &\longrightarrow \forall r.(E \sqcap F), \\ \forall r.\top &\longrightarrow \top. \end{aligned}$$

A concept description C *is called* normalized *if none of the normalization rules can be applied to some part of* C.

The rules should be read modulo commutativity of conjunction; e.g., $\exists r.E \sqcap \forall r.F$ is also normalized to $\exists r.(E \sqcap F) \sqcap \forall r.F$. An unnormalized concept description C can be normalized by exhaustively applying the normalization rules in C. The resulting (normalized) concept description is called *normal form of* C. Since each normalization rule preserves equivalence, the normal form of C is equivalent to C. The $\mathcal{ALE}$-description tree corresponding to the normal form of C is denoted by $\mathcal{G}_C$.

If only the rule $\forall r.\top \longrightarrow \top$ is exhaustively applied to a concept description C, then the resulting concept description is called $\top$*-normal form of* C, and the corresponding tree is denoted by $\mathcal{G}_C^\top$.

Now, subsumption can be characterized in terms of homomorphisms as follows:

Lemma 16 *(Baader et al., 1999b) Let* C, D *be* $\mathcal{ALE}$*-concept descriptions. Then,* $C \sqsubseteq D$ *iff there exists a homomorphism from* $\mathcal{G}_D^\top$ *to* $\mathcal{G}_C$.

It should be noted that the theorem stated in (Baader et al., 1999b) requires a homomorphism from $\mathcal{G}_D$ instead of $\mathcal{G}_D^\top$. However, a closer look at the proof in (Baader et al., 1998) reveals that $\top$-normalization of the subsumer is sufficient. This will be important in the following when we use Lemma 16 as basis for the $\mathcal{ALE}$-matching algorithm. In this context, the subsumer is an instantiation of the pattern D, and thus, normalizing the subsumer $\sigma(D)$ depends on the substitution σ, which is not known in advance. Therefore, it is crucial that only very simple normalizations of the subsumer are required.

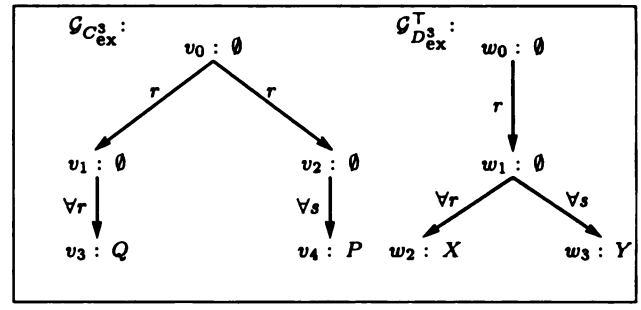


Figure 5: The description trees for C^3_{ex} and D^3_{ex}.

The $\mathcal{ALE}$-matching algorithm

Following Lemma 16, the $\mathcal{EL}$-matching algorithm is modified as follows: (i) instead of the trees $\mathcal{G}(C)$ and $\mathcal{G}(D)$, we now consider $\mathcal{G}_C$ and $\mathcal{G}_D^\top$, where the $\top$-normal form of D is obtained by treating concept variables like concept names; (ii) homomorphisms are computed with respect to Definition 14, where again variables are ignored in 2.

This straightforward extension of the $\mathcal{EL}$-matching algorithm is sufficient to solve the $\mathcal{ALE}$-matching problem $C^2_{\text{ex}} \sqsubseteq D^2_{\text{ex}}$. There exists exactly one homomorphism φ from $\mathcal{G}^\top_{D^2_{\text{ex}}}$ into $\mathcal{G}_{C^2_{\text{ex}}}$ (see Figure 4). Following the $\mathcal{EL}$-matching algorithm, φ gives rise to the matcher σ with $\sigma(X) := lcs\{\bot, \neg P\} \equiv \neg P$. The singleton set $\{\sigma\}$ is indeed an s-complete set.

However, as illustrated by the next example, this simple extension of the $\mathcal{EL}$-matching algorithm does not work in general.

Example 17 *Consider the* $\mathcal{ALE}$*-matching problem* $C^3_{ex} \sqsubseteq^? D^3_{ex}$, *where*

$$\begin{aligned} C^3_{ex} &:= (\exists r.\forall r.Q) \sqcap (\exists r.\forall s.P) \\ D^3_{ex} &:= \exists r.(\forall r.X \sqcap \forall s.Y). \end{aligned}$$

The description trees corresponding to C^3_{ex} *and* D^3_{ex} *are depicted in Figure 5. Obviously,* $\sigma := \{X \mapsto Q, Y \mapsto \top\}$ *and* $\tau := \{X \mapsto \top, Y \mapsto P\}$ *are solutions of the matching problem. However, there is no homomorphism from* $\mathcal{G}^\top_{D^3_{ex}}$ *into* $\mathcal{G}_{C^3_{ex}}$. *Indeed, the node* w_1 *can be mapped either on* v_1 *or on* v_2. *In the former case,* w_2 *can be mapped on* v_3, *but then there is no way to map* w_3. *In the latter case,* w_3 *must be mapped on* v_4, *but then there is no node* w_2 *can be mapped on.*

The problem is that Lemma 16 requires the subsumer to be in $\top$-normal form. However, the $\top$-normal form of the instantiated concept pattern depends on the matcher, and thus cannot be computed in advance. For instance, in Example 17 the instances $\sigma(D^3_{\text{ex}})$ and

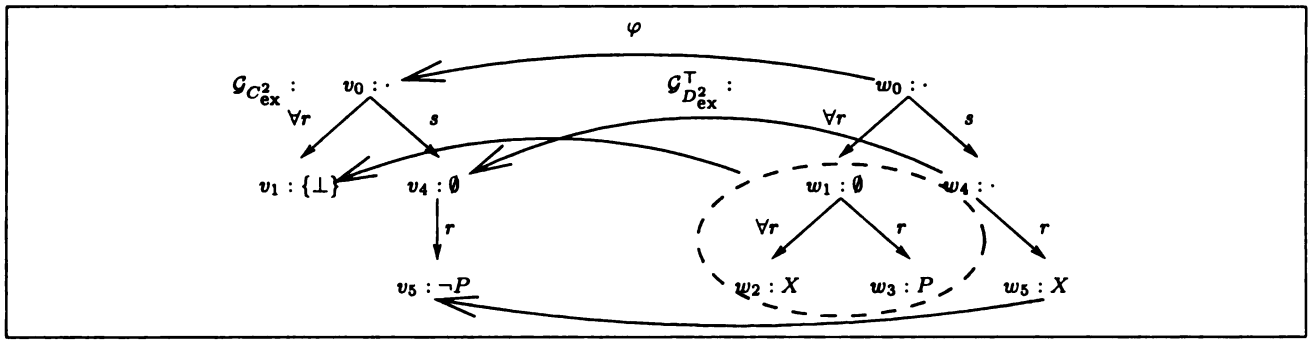


Figure 4: Subsumption for $\mathcal{ALE}$.

$\tau(D^3_{ex})$ of D^3_{ex} are not $\top$-normalized since they contain $\forall s.\top$ and $\forall r.\top$, respectively. The description tree for the $\top$-normalized concept description $\sigma(D^3_{ex})$ does not include the node w_3 and the $\forall s$-edge leading to it. Analogously, for $\tau(D)$, w_2 would be deleted.

This illustrates that the instances of a pattern are not necessarily in $\top$-normal form and that the $\top$-normal form depends on the particular instance. However, only those matchers cause problems that replace variables by the top-concept. Therefore, instead of considering homomorphisms originating from $\mathcal{G}^{\top}_D$, the algorithm computes homomorphisms from the so-called $\top$-patterns of D into $\mathcal{G}_C$.

Definition 18 *The concept pattern E is called* $\top$-pattern *of D iff E is obtained from D by replacing some of the variables in D by $\top$.*

In our example, we obtain the following $\top$-normalized $\top$-patterns for D^3_{ex}: D^3_{ex}, $\exists r.(\forall r.X)$, $\exists r.(\forall s.Y)$, and $\exists r.\top$. Matching these patterns against C^3_{ex}, the extended matching algorithm described above computes the following sets of solutions: $\emptyset$, $\{\sigma\}$, $\{\tau\}$, and $\{\{X \mapsto \top, Y \mapsto \top\}\}$. The union of these sets provides us with an s-complete set of solutions of $C^3_{ex} \sqsubseteq^? D^3_{ex}$.

The matching algorithm for $\mathcal{ALE}$ obtained from these considerations is depicted in Figure 6. Here $\ell(v)$ denotes the label of v in $\mathcal{G}^{\top}_E$, and $C^{\varphi(v)}$ stands for the concept description corresponding to the subtree of $\mathcal{G}_C$ with root $\varphi(v)$.

5.3 Complexity of computing i-complete sets

There are two different aspects to consider. First, the size of i-complete (and s-complete) sets, and second, the complexity of the algorithm for computing these sets. The following theorem shows that the size of complete sets may grow exponentially in the size of the matching problem.

Input: $\mathcal{ALE}$-matching problem $C \sqsubseteq^? D$
Output: s-complete set $\mathcal{C}$ for $C \sqsubseteq^? D$

$\mathcal{C} := \emptyset$
For all $\top$-patterns E of D do
  For all homomorphisms φ from $\mathcal{G}^{\top}_E$ into $\mathcal{G}_C$
    Define τ by
      $\tau(X) := lcs\{C^{\varphi(v)} \mid X \in \ell(v)\}$
        for all X in E and
      $\tau(X) := \top$
        for all X in D not contained in E
    $\mathcal{C} := \mathcal{C} \cup \{\tau\}$

Figure 6: The $\mathcal{ALE}$-matching algorithm.

Theorem 19 *Both, for $\mathcal{EL}$ and $\mathcal{ALE}$*

1. *the cardinality of minimal i-complete and s-complete sets of matchers and the size of the matchers in these sets are at most exponential in the size of the matching problem;*
2. *these exponential upper-bounds are tight.*

We illustrate the second part of the theorem by two examples, one for the cardinality of complete sets and one for the size of the matchers.

Example 20 *Let C_n be the $\mathcal{EL}$-/$\mathcal{ALE}$-concept description*

$$C_n := \prod_{i=1}^{n} \exists r.(\prod_{j=1}^{n} \exists r.(A_i \sqcap B_j),$$

and D_n be the $\mathcal{EL}$-/$\mathcal{ALE}$-concept pattern

$$D_n := \prod_{i=1}^{n} \exists r.\exists r.X_i.$$

For the matching problem $C_n \sqsubseteq^? D_n$ and a word $w = a_1 \cdots a_n \in \{1,\ldots,n\}^n$ of length n over the alphabet $\{1,\ldots,n\}$, the substitution $\sigma_w(X_i) := A_i \sqcap B_j$ for $a_i = j$ is obviously an i-minimal and s-minimal matcher.

Furthermore, for different words $w, w' \in \{1, \ldots, n\}^n$, one obtains i-incomparable and s-incomparable matchers. Since there are n^n such words, the number of i-minimal and s-minimal matchers grows exponentially in n, and thus in the size of the matching problems $C_n \sqsubseteq^? D_n$.

The next example demonstrates that the size of a single matcher in an s-complete/i-complete set may grow exponentially in the size of the matching problem.

Example 21 *In (Baader et al., 1999b), it has been shown that there is a sequence $E_1, E_2, \ldots$ of $\mathcal{EL}$-/$\mathcal{ALE}$-concept descriptions such that the size of $lcs(E_1, \ldots, E_n)$ grows exponentially in the size of $E_1, \ldots, E_n$.*

Now, consider the $\mathcal{EL}$-/$\mathcal{ALE}$-matching problem $C'_n \sqsubseteq^? D'_n$, where $C'_n := \exists r_1.E_1 \sqcap \cdots \sqcap \exists r_n.E_n$ and $D'_n := \exists r_1.X \sqcap \cdots \sqcap \exists r_n.X$. Clearly, for an i-minimal or s-minimal matcher σ of this matching problem, $\sigma(X) \equiv lcs(E_1, \ldots, E_n)$. Thus, $\sigma(X)$ grows exponentially in the size of the matching problem.

We now turn to the complexity of the $\mathcal{ALE}$-matching algorithm and show that it runs in exponential time. Obviously, this also implies the exponential upper-bound stated in the first part of Theorem 19. Similar arguments can be employed for the $\mathcal{EL}$-matching algorithm.

First, note that the number of $\top$-patterns E of D is at most exponential in the size of D. Also, the size of $\mathcal{G}_E^\top$ for each such $\top$-pattern is linear in the size of D. As shown in (Baader et al., 1999b), the size of $\mathcal{G}_C$ is at most exponential in the size of C. Consequently, it is easy to see that the number of homomorphisms from $\mathcal{G}_E^\top$ into $\mathcal{G}_C$ is at most exponential in the size of C and D. This shows that the number of matchers σ computed by the algorithm is at most exponential in the size of the given matching problem.

It remains to be shown that each such σ can be computed in exponential time. As proved in (Baader et al., 1999b), the lcs of n concepts $C_1, \ldots, C_n$ can be computed in time bounded by the product of the sizes of the concepts C_i. Since the size of $\mathcal{G}_C$, and thus also of $C^{\varphi(v)}$, is at most exponential in C and the number of nodes v in $\mathcal{G}_E^\top$ is linear in D, it follows that each $\sigma(X)$ can be computed in time exponential in the size of C and D. Finally, the fact that D (and thus also E) contains only a linear number of variables shows that σ can be computed in exponential time. This proves

Corollary 22 *Computing i- and s-complete sets for $\mathcal{EL}$-/$\mathcal{ALE}$-matching problems modulo subsumption can be carried out in time exponential in the size of the matching problem.*

Recall that we are actually interested in computing a *minimal* i-complete set of the matching problem $C \sqsubseteq^? D$. As mentioned at the beginning of this section, given an s-complete set, one can compute a minimal i-complete set by testing subsumption between the instances $\tau(D)$ of D, where τ belongs to the s-complete set. Since the size of these instances may be exponential in the size of the matching problem, and since subsumption is polynomial in $\mathcal{EL}$ and NP-complete in $\mathcal{ALE}$, we obtain the following complexity result for computing minimal i-complete sets:

Corollary 23 *Computing minimal i-complete sets for matching modulo subsumption can be carried out in exponential time for $\mathcal{EL}$- and in exponential space for $\mathcal{ALE}$-matching problems.*

6 Computing d-minimal matchers

In order to realize the second step of the matching algorithm sketched at the end of Section 4, we must show how to compute all d-minimal (i.e., reduced) matchers up to s-equivalence of a given matching problem modulo *equivalence*. For such a matching problem $C \equiv^? D$, sets containing at least all d-minimal matchers up to s-equivalence are called *d-complete*. Such a set is called *minimal* if it contains exactly one d-minimal matcher for each s-equivalence class.

The following theorem implies that, in the worst case, algorithms computing d-complete sets need exponential time.

Theorem 24 *Let $C \equiv^? D$ be an $\mathcal{EL}$- or $\mathcal{ALE}$-matching problem modulo equivalence. Then,*

1. *the cardinality of (minimal) d-complete sets can grow exponentially in the size of the matching problem;*
2. *however, there always exists a (minimal) d-complete set such that the size of each matcher in this set is polynomially bounded.*

The first statement of the theorem is an easy consequence of the fact that every d-complete set for the $\mathcal{EL}$-/$\mathcal{ALE}$-matching problem $\exists r.A_1 \sqcap \cdots \sqcap \exists r.A_n \equiv^? \exists r.X_1 \sqcap \cdots \exists r.X_n$ contains at least an exponential number of matchers.

The non-trivial result to prove is the second part of Theorem 24 (see (Baader and Küsters, 1999)). This result yields a naïve exponential-time algorithm for

computing d-complete sets: Enumerate all substitutions up to the polynomial bound and filter out those that are not solutions of the problem or that are not d-minimal. The filtering can be realized by a polynomial time algorithm using an oracle for subsumption. Obviously, this naïve algorithm is very inefficient.

For $\mathcal{EL}$, we will sketch an improved exponential time algorithm, which significantly prunes the search space for candidate matchers. This algorithm is based on the following (non-trivial) lemma:

Lemma 25 *(Baader and Küsters, 1999) The matcher σ of the $\mathcal{EL}$-matching problem $C \equiv^? D$ is d-minimal iff it is s-maximal and $\sigma(X)$ is reduced for all variables X.*

Here s-maximality of matchers is defined w.r.t. $\sqsubseteq_s$ in the obvious way. Note that this lemma does not hold for $\mathcal{ALE}$, and thus it is not clear how to extend our approach from $\mathcal{EL}$ to $\mathcal{ALE}$.

Because of the lemma, the task of computing a d-complete set of matchers in $\mathcal{EL}$ can be split into two subtasks. First, compute an *s-co-complete* set of matchers, i.e., a set containing all s-maximal matchers up to s-equivalence. Second, for every matcher σ in the set and every variable X, compute a reduced concept description equivalent to $\sigma(X)$. In (Baader and Küsters, 1999), it is shown that, for $\mathcal{EL}$, the second task can be realized by a polynomial time algorithm. In the following, we sketch an algorithm for performing the first task (see (Baader and Küsters, 1999) for a complete description and proof of correctness). Roughly speaking, this algorithm is the dual of the one in Figure 2. The duality occurs at two places in the algorithm.

First, instead of computing substitutions τ satisfying $C \sqsubseteq \tau(D)$, we now compute substitutions τ that satisfy $C \sqsupseteq \tau(D)$. To make sure that the substitutions computed by the algorithm really solve the matching problem $C \equiv^? D$, we use the subsumption algorithm for $\mathcal{EL}$ to remove those substitutions not satisfying $C \sqsubseteq \tau(D)$. In order to obtain substitutions τ satisfying $C \sqsupseteq \tau(D)$, we now consider homomorphisms in the other direction, i.e., from $\mathcal{G}(C)$ into $\mathcal{G}(D)$. To be more precise, we consider homomorphisms ψ that are *partial* in the following sense: (i) certain nodes of $\mathcal{G}(C)$ need not be mapped onto nodes of $\mathcal{G}(D)$; and (ii) for certain nodes the inclusion condition between labels need not hold. The idea is that the parts of C that are not mapped and the labels violating the inclusion condition are covered by the concepts substituted for the variables. For this reason, a partial homomorphism implicitly associates with each variable a set of concepts that must be covered by this variable. (Note that, for a given partial homomorphism, there are different ways of associating concepts with variables.)

The second duality occurs when defining the substitution τ: in the definition of $\tau(X)$, the lcs is replaced by conjunction of the concepts associated with X. The exact definition of partial homomorphisms is such that a given partial homomorphism can always be extended to a (total) homomorphism φ from $\mathcal{G}(C)$ into $\mathcal{G}(\tau(D))$ for the constructed substitution τ. A detailed description of the algorithm can be found in (Baader and Küsters, 1999). Here, we only illustrate it by the matching problem $C^1_{\mathrm{ex}} \equiv^? D^1_{\mathrm{ex}}$. For instance, $\psi := \{v_0 \mapsto w_0, v_1 \mapsto w_1, v_2 \mapsto w_2, v_3 \mapsto w_3\}$ is a partial homomorphism from $\mathcal{G}(C^1_{\mathrm{ex}})$ into $\mathcal{G}(D^1_{\mathrm{ex}})$. Here the nodes v_4 and v_5 are not mapped by ψ. The "missing parts" can be covered by associating with X the concepts W and $\exists \mathrm{hc}.(\mathsf{W} \sqcap \mathsf{P})$, and with Y the concept D. In addition, it is not hard to verify that the substitution $\tau := \{X \mapsto \mathsf{W} \sqcap \exists \mathrm{hc}.(\mathsf{W} \sqcap \mathsf{P}), Y \mapsto \mathsf{D}\}$ also satisfies $C^1_{\mathrm{ex}} \sqsubseteq \tau(D^1_{\mathrm{ex}})$. Thus, τ is a matcher in the computed s-co-complete set.

A sketch of the proof of Lemma 4

It is easy to see that, by construction, the matchers in the s-co-complete set computed by the algorithm sketched above are of size polynomial in the size of the matching problem. This shows Lemma 4 for $\mathcal{EL}$, since an s-co-complete set can only be empty if the matching problem is not solvable.

For $\mathcal{ALE}$, things are more complicated. In the sequel, let σ' be an arbitrary matcher of the $\mathcal{ALE}$-matching problem $C \equiv^? D$ (where, without loss of generality, $\sigma'(X)$ is in $\forall$-normal form for every variable X). The task is to construct from σ' a new matcher σ of size polynomially bounded in the size of the matching problem. In the following, let C^r be the reduced concept description equivalent to C. Note that the size of C^r is linear in the size of C. Furthermore, let E' be the $\forall$-normal form of $\sigma'(D)$.

First, assume that C^r does not contain $\bot$. As an easy consequence of Lemma 7, we can show that there exists an injective homomorphism ψ from $\mathcal{G}(C^r)$ into $\mathcal{G}(E')$. Let $\mathcal{G}'$ be the image of $\mathcal{G}(C^r)$ under ψ. In principle, σ is obtained from σ' by removing from each description $\sigma'(X)$ all the parts that are not needed to obtain $\mathcal{G}'$. Consequently, the size of $\sigma(X)$ is bounded by the size of C^r, and thus σ is polynomial in the size of the matching problem.

It remains to be shown that σ solves the matching problem $C \equiv^? D$. Let E be the $\forall$-normal form of $\sigma(D)$.

The construction of σ ensures that ψ is still an injective homomorphism from $\mathcal{G}(C^r)$ into $\mathcal{G}(E)$. This implies that $\sigma(D) \equiv E \sqsubseteq C^r \equiv C$. Conversely, the definition of σ also implies $\sigma' \sqsubseteq_s \sigma$, and thus $C \equiv \sigma'(D) \sqsubseteq \sigma(D)$.

If C^r contains $\bot$, then each description $\sigma(X)$ must be extended by those parts of $\mathcal{G}(E')$ that contribute to inconsistencies. In (Baader and Küsters, 1999), these parts are identified and it is shown that they can be chosen such that the size of σ can still be polynomially bounded by the size of the matching problem.

7 Future work

The remaining technical challenge is to design practical algorithms for computing d-minimal matchers for $\mathcal{ACE}$, and for $\mathcal{ACN}$ and its extension to the CLASSIC description language.

We will also evaluate the usefulness of matching for removing redundancies in knowledge bases within our process engineering application (Baader and Sattler, 1996). In order to apply matching to the problem of integrating knowledge bases (Borgida and Küsters, 1999), we first need to extend the matching algorithm to an algorithm that takes schemas (i.e., certain types of inclusion axioms) into account.

References

Baader, F. and Küsters, R. (1999). Matching in Description Logics with Existential Restrictions Revisited. Tech. Report LTCS-Report 99-13, LuFg Theoretical Computer Science, RWTH Aachen, Germany. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

Baader, F., Küsters, R., Borgida, A., and McGuinness, D. (1999a). Matching in description logics. *Journal of Logic and Computation*, 9(3):411–447.

Baader, F., Küsters, R., and Molitor, R. (1998). Computing Least Common Subsumer in Description Logics with Existential Restrictions. Tech. Report LTCS-Report 98-09, LuFg Theoretical Computer Science, RWTH Aachen, Germany. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

Baader, F., Küsters, R., and Molitor, R. (1999b). Computing least common subsumer in description logics with existential restrictions. In Dean, T., editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 96–101. Morgan Kaufmann.

Baader, F. and Narendran, P. (1998). Unification of concept terms in description logics. In *Proceedings of the 13th Biennial European Conference on Artificial Intelligence (ECAI-98)*. Brighton, UK.

Baader, F. and Sattler, U. (1996). Knowledge representation in process engineering. In *Proceedings of the International Workshop on Description Logics*, Cambridge (Boston), MA, U.S.A. AAAI Press/The MIT Press.

Borgida, A., Brachman, R. J., McGuinness, D. L., and Resnick, L. A. (1989). CLASSIC: A structural data model for objects. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pages 59–67, Portland, OR.

Borgida, A. and Küsters, R. (1999). What's not in a name? Initial explorations of a structural approach to integrating large concept knowledge-bases. Technical Report DCS-TR-391, Rutgers University, USA. Available via ftp://ftp.cs.rutgers.edu/pub/technical-reports/.

Borgida, A. and McGuinness, D. L. (1996). Asking queries about frames. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, pages 340–349, San Francisco, Calif. Morgan Kaufmann.

Brachman, R. J., McGuinness, D. L., Patel-Schneider, P. F., Resnick, L. A., and Borgida, A. (1991). Living with CLASSIC: When and how to use a KL-ONE-like language. In Sowa, J., editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, San Mateo, Calif.

Cohen, W., Borgida, A., and Hirsh, H. (1992). Computing least common subsumers in description logics. In Swartout, W., editor, *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 754–760, San Jose, CA. MIT Press.

Donini, F., Hollunder, B., Lenzerini, M., Marchetti, A., Nardi, D., and Nutt, W. (1992). The complexity of existential quantification in concept languages. *Artificial Intelligence*, 2–3:309–327.

McGuinness, D. (1996). *Explaining Reasoning in Description Logics*. PhD thesis, Department of Computer Science, Rutgers University. Also available as a Rutgers Technical Report LCSR-TR-277.

Expressive ABox Reasoning with Number Restrictions, Role Hierarchies, and Transitively Closed Roles

Volker Haarslev and **Ralf Möller**
University of Hamburg, Computer Science Department,
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
http://kogs-www.informatik.uni-hamburg.de/~<name>/

Abstract

We present a new tableaux calculus deciding the ABox consistency problem for the expressive description logic $\mathcal{ALCNH}_{R+}$. Prominent language features of $\mathcal{ALCNH}_{R+}$ are number restrictions, role hierarchies, transitively closed roles, and generalized concept inclusions. The ABox description logic system RACE is based on the calculus for $\mathcal{ALCNH}_{R+}$.

1 Introduction

Experiences with concept languages indicate that at least description logics (DLs) with negation and disjunction are required to solve practical modeling problems without resorting to ad hoc extensions. The requirements derived from practical applications of DLs ask for even more expressive languages. For instance, in [14] the need for transitive roles is demonstrated for representing part-whole relations, family relations or partial orders in general. It is argued that the trade-off between expressivity and complexity favors the integration of transitively closed roles instead of a transitive closure operator for roles. Other examples are given in [8], where the area of medical terminology is discussed. Design studies for the Galen project identified the need for modeling of transitive part-whole, causal and compositional relations, and to organize these relations into a hierarchy. Moreover, generalized concept inclusions were also required as a modeling tool, e.g. for expressing sufficient conditions of concepts.

2 The Description Logic $\mathcal{ALCNH}_{R+}$

Motivated by the above-mentioned requirements we introduce in this paper an ABox tableaux calculus for the description logic $\mathcal{ALCNH}_{R+}$. It augments the basic logic $\mathcal{ALC}$ [15] with number restrictions, role hierarchies, and transitively closed roles. Note that these language features imply the presence of generalized concept inclusions and cyclic concepts. The use of number restrictions in combination with transitive roles and role hierarchies is syntactically restricted: no number restrictions are possible for (i) transitive roles and (ii) for any role which has a transitive subrole. Furthermore, we assume that the unique name assumption holds for ABox individuals.

$\mathcal{ALCNH}_{R+}$ is an extension of $\mathcal{ALCNH}$ that itself can be polynomially reduced to $\mathcal{ALCNR}$ [1] and vice versa. It is possible to rephrase every hierarchy of role names with a set of role conjunctions and vice versa [1]. Thus, our work on $\mathcal{ALCNH}_{R+}$ extends the work on $\mathcal{ALCNR}$ by additionally providing transitively closed roles. $\mathcal{ALCNH}_{R+}$ also extends other related description logics such as $\mathcal{ALC}_{R+}$ [14] and $\mathcal{ALCH}f_{R+}$ [8]. Recently, the work on these logics has been extended and a tableaux calculus for deciding concept consistency for the language $\mathcal{ALCQHI}_{R+}$ has been presented in [11]. Another approach is presented in [2] where the logic $\mathcal{CIQ}$ for reasoning with TBoxes and ABoxes is introduced. In comparison to $\mathcal{ALCNH}_{R+}$ and the other approaches mentioned above $\mathcal{CIQ}$ offers more operators (e.g. the transitive closure) but does not support role hierarchies and allows number restrictions only for primitive roles.

ABox reasoning truly extends the usefulness of description logics in practical applications. The increase of expressiveness is also reflected in an increase of the complexity of the tableaux rules (see Section 4.1 for more details). An alternative might be the so-called "precompletion approach" originally developed for the language $\mathcal{ALCQ}$ [7] and recently adapted to $\mathcal{ALCH}_{R+}$ [16]. The idea behind the precompletion approach is to transform given ABoxes in a way such that ABox satisfiability is reduced to concept satisfiability. How-

ever, there currently exist no calculi for computing the precompletion of ABoxes for languages such as $\mathcal{ALCNH}_{R^+}$ or even $\mathcal{ALCQHI}_{R^+}$.

2.1 The Concept Language

We present the syntax and semantics of the language for specifying concept and role inclusions.

Definition 1 (Role Inclusions, Role Hierarchy) Let P and T be disjoint sets of non-transitive and transitive *role* names, respectively, and let R be defined as $R = P \cup T$. Let R and S be role names, then $\mathsf{R} \sqsubseteq \mathsf{S}$ *(role inclusion axiom)* is a terminological axiom. Given a set of role inclusion axioms, we define a *role hierarchy* where $\sqsubseteq^*$ is the reflexive transitive closure of $\sqsubseteq$ over R.

Additionally we define the set of ancestors and descendants of a role.

Definition 2 (Role Descendants/Ancestors) Given a role hierarchy the set $\mathsf{R}^{\uparrow} := \{\mathsf{S} \in R \mid \mathsf{R} \sqsubseteq^* \mathsf{S}\}$ defines the *ancestors* and $\mathsf{R}^{\downarrow} := \{\mathsf{S} \in R \mid \mathsf{S} \sqsubseteq^* \mathsf{R}\}$ the *descendants* of a role R. We also define the set $S := \{\mathsf{R} \in P \mid \mathsf{R}^{\downarrow} \cap T = \emptyset\}$ of *simple* roles that are neither transitive nor have a transitive role as descendant.

Definition 3 (Concept Terms) Let C be a set of concept names which is disjoint from R. Any element of C is a *concept term*. If C and D are concept terms, $\mathsf{R} \in R$ is an arbitrary role, $\mathsf{S} \in S$ is a simple role, $n > 1$, and $m > 0$, then the following expressions are also concept terms:

- $\top$ *(top concept)*,
- $\bot$ *(bottom concept)*,
- $\mathsf{C} \sqcap \mathsf{D}$ *(conjunction)*,
- $\mathsf{C} \sqcup \mathsf{D}$ *(disjunction)*,
- $\neg\mathsf{C}$ *(negation)*,
- $\forall \mathsf{R}.\mathsf{C}$ *(concept value restriction)*,
- $\exists \mathsf{R}.\mathsf{C}$ *(concept exists restriction)*,
- $\exists_{\leq m}\mathsf{S}$ *(at most number restriction)*,
- $\exists_{\geq n}\mathsf{S}$ *(at least number restriction)*.

For an arbitrary role R, the term $\exists_{\geq 1}\mathsf{R}$ can be rewritten as $\exists \mathsf{R}.\top$, $\exists_{\geq 0}\mathsf{R}$ as $\top$, and $\exists_{\leq 0}\mathsf{R}$ as $\forall \mathsf{R}.\bot$. Thus, we do not consider these terms as number restrictions in our language.

The concept language is syntactically restricting the combination of number restrictions and transitive roles. Number restrictions are only allowed for *simple* roles. This restriction is motivated by doubtful semantics for an unrestricted combinability and a simplified tableaux decision procedure. Moreover, this decision is supported by a recent undecidability result for the logic $\mathcal{ALCHNI}_{R^+}$ in case of an unrestricted combinability [11].

Definition 4 (Generalized Concept Inclusions) If C and D are concept terms, then $\mathsf{C} \sqsubseteq \mathsf{D}$ *(generalized concept inclusion* or *GCI)* is a terminological axiom as well.

A finite set of terminological axioms $\mathcal{T}$ is called a *terminology* or *TBox*. GCIs can be used to represent terminological cycles. There exist at least two ways to deal with GCIs in a tableaux calculus. The 'internalization' approach (e.g. see in [9]) makes use of the fact that the expressiveness of GCIs is already implied by the combination of role hierarchies and transitive roles. However, with the presence of arbitrary ABoxes one has also to consider unrelated individuals. Therefore, we pursue a different and more direct approach that extends an ABox tableaux calculus by new constructs and rules directly dealing with GCIs (see Definition 7).

The next definition gives a set-theoretic semantics to the language introduced above.

Definition 5 (Semantics) An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$ (the domain) and an interpretation function $\cdot^{\mathcal{I}}$. The interpretation function maps each concept name C to a subset $\mathsf{C}^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each role name R to a subset $\mathsf{R}^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Let the symbols C, D be concept expressions, and R, S be role names. Then the interpretation function can be extended to arbitrary concept and role terms as follows ($\|\cdot\|$ denotes the cardinality of a set):

$$
\begin{aligned}
(\mathsf{C} \sqcap \mathsf{D})^{\mathcal{I}} &:= \mathsf{C}^{\mathcal{I}} \cap \mathsf{D}^{\mathcal{I}} \\
(\mathsf{C} \sqcup \mathsf{D})^{\mathcal{I}} &:= \mathsf{C}^{\mathcal{I}} \cup \mathsf{D}^{\mathcal{I}} \\
(\neg\mathsf{C})^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \setminus \mathsf{C}^{\mathcal{I}} \\
(\exists \mathsf{R}.\mathsf{C})^{\mathcal{I}} &:= \{a \in \Delta^{\mathcal{I}} \mid \exists b \in \Delta^{\mathcal{I}} : (a,b) \in \mathsf{R}^{\mathcal{I}}, b \in \mathsf{C}^{\mathcal{I}}\} \\
(\forall \mathsf{R}.\mathsf{C})^{\mathcal{I}} &:= \{a \in \Delta^{\mathcal{I}} \mid \forall b \in \Delta^{\mathcal{I}} : (a,b) \in \mathsf{R}^{\mathcal{I}} \Rightarrow b \in \mathsf{C}^{\mathcal{I}}\} \\
(\exists_{\geq n}\mathsf{R})^{\mathcal{I}} &:= \{a \in \Delta^{\mathcal{I}} \mid \|\{b \mid (a,b) \in \mathsf{R}^{\mathcal{I}}\}\| \geq n\} \\
(\exists_{\leq n}\mathsf{R})^{\mathcal{I}} &:= \{a \in \Delta^{\mathcal{I}} \mid \|\{b \mid (a,b) \in \mathsf{R}^{\mathcal{I}}\}\| \leq n\}
\end{aligned}
$$

An interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ iff it satisfies (1) $\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$ for all terminological axioms (GCIs) $\mathsf{C} \sqsubseteq \mathsf{D}$ in $\mathcal{T}$ and $\mathsf{R}^{\mathcal{I}} \subseteq \mathsf{S}^{\mathcal{I}}$ for all terminological axioms $\mathsf{R} \sqsubseteq \mathsf{S}$ (role inclusions) in $\mathcal{T}$, and (2) iff for every

$R \in T : R^{\mathcal{I}} = (R^{\mathcal{I}})^+$. A concept term C *subsumes* a concept term D w.r.t. a TBox $\mathcal{T}$ (written $D \preceq_{\mathcal{T}} C$), iff $D^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$. A concept term C is *satisfiable* w.r.t. a TBox $\mathcal{T}$ iff there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$.

One of the basic reasoning services for a description logic formalism is computing the subsumption relationship for atomic concepts. This inference is needed in the TBox to build a hierarchy of concept names w.r.t. specificity. Satisfiability and subsumption can be mutually reduced to each other since $C \preceq_{\mathcal{T}} D$ iff $C \sqcap \neg D$ is not satisfiable w.r.t. $\mathcal{T}$ and C is unsatisfiable w.r.t. $\mathcal{T}$ iff $C \preceq_{\mathcal{T}} \bot$.

2.2 The Assertional Language

In the following, the language for representing knowledge about individual worlds is introduced. An *ABox* $\mathcal{A}$ is a finite set of assertional axioms which are defined as follows:

Definition 6 (ABox Assertions)
Let $O = O_O \cup O_N$ be a set of individual names, where the set O_O of "old" names is disjoint with the set O_N of "new" names. If C is a concept term, R a role name, and $a, b \in O$ are individual names, then the following expressions are *assertional axioms*:

- $a : C$ *(concept assertion)*,
- $(a, b) : R$ *(role assertion)*.

The interpretation function $\cdot^{\mathcal{I}}$ of the interpretation $\mathcal{I}$ for the concept language can be extended to the assertional language by additionally mapping every individual name from O to a single element of $\Delta^{\mathcal{I}}$ in a way such that for $a, b \in O_O$, $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$ *(unique name assumption)*. This ensures that different individuals in O_O are interpreted as different objects. The unique name assumption does not hold for elements of O_N, i.e. for $a, b \in O_N$, $a^{\mathcal{I}} = b^{\mathcal{I}}$ may hold even if $a \neq b$. An interpretation satisfies an assertional axiom $a : C$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $(a, b) : R$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

An interpretation is a *model* of an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ iff it is a model of $\mathcal{T}$ and furthermore satisfies all assertional axioms in $\mathcal{A}$. An ABox is *consistent* w.r.t. a TBox $\mathcal{T}$ iff it has a model w.r.t. $\mathcal{T}$. An individual b is called a *direct successor* of an individual a in an ABox $\mathcal{A}$ iff $\mathcal{A}$ contains the assertional axiom $(a, b) : R$. An individual b is called a *successor* of a if it is either a direct successor of a or there exists in $\mathcal{A}$ a chain of assertions $(a, b_1) : R_1, (b_1, b_2) : R_2, \ldots, (b_n, b) : R_{n-1}$. In case that $R_i = R_j$ or $R_i \in R^{\downarrow}$ for all $i, j \in 1..n-1$ we call b the (direct) R-*successor* of a. A (direct) *predecessor* is defined analogously. An individual a is called an *instance* of a concept term C in an interpretation $\mathcal{I}$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$. The *direct types* of an individual are the most specific atomic concepts which the individual is an instance of.

The ABox consistency problem is to decide whether a given ABox $\mathcal{A}$ is consistent w.r.t. a TBox $\mathcal{T}$. Satisfiability of concept terms can be reduced to ABox consistency as follows: A concept term C is satisfiable iff the ABox $\{a : C\}$ is consistent. *Instance checking* tests whether an individual a is an instance of a concept term C w.r.t. an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$, i.e. whether $\mathcal{A}$ entails $a : C$ w.r.t. $\mathcal{T}$. This problem is reduced to the problem of deciding if the ABox $\mathcal{A} \cup \{a : \neg C\}$ is inconsistent.

3 An ABox Example

Before we continue with the calculus for $\mathcal{ALCNH}_{R^+}$, we illustrate in the following the expressiveness of $\mathcal{ALCNH}_{R^+}$ with a TBox and ABox example about family relationships. This example uses prominent features of $\mathcal{ALCNH}_{R^+}$ such as transitive roles, role hierarchies, number restrictions and generalized concept inclusions.

In the TBox *family* we assume a role has_descendant which is declared to be *transitive*, has_gender which is declared as a feature (e.g. this can be achieved by adding the axiom $\top \sqsubseteq \exists_{\leq 1}$ has_gender), and a role has_sibling. The TBox *family* contains the following role axioms.

has_child $\sqsubseteq$ has_descendant

has_sister $\sqsubseteq$ has_sibling

has_brother $\sqsubseteq$ has_sibling

The TBox *family* contains concept axioms specifying the domain and/or range of the roles introduced above (the domain A of a role R can be expressed by the axiom $\exists_{\geq 1} R \sqsubseteq A$ and the range B by $\top \sqsubseteq \forall R . B$).

$\exists_{\geq 1}$ has_descendant $\sqsubseteq$ human

$\top \sqsubseteq \forall$ has_descendant . human

$\exists_{\geq 1}$ has_child $\sqsubseteq$ parent

$\exists_{\geq 1}$ has_sibling $\sqsubseteq$ sibling

$\top \sqsubseteq \forall$ has_sibling . sibling

$\top \sqsubseteq \forall$ has_sister . sister

$\top \sqsubseteq \forall$ has_brother . brother

$\top \sqsubseteq \forall$ has_gender . (female $\sqcup$ male)

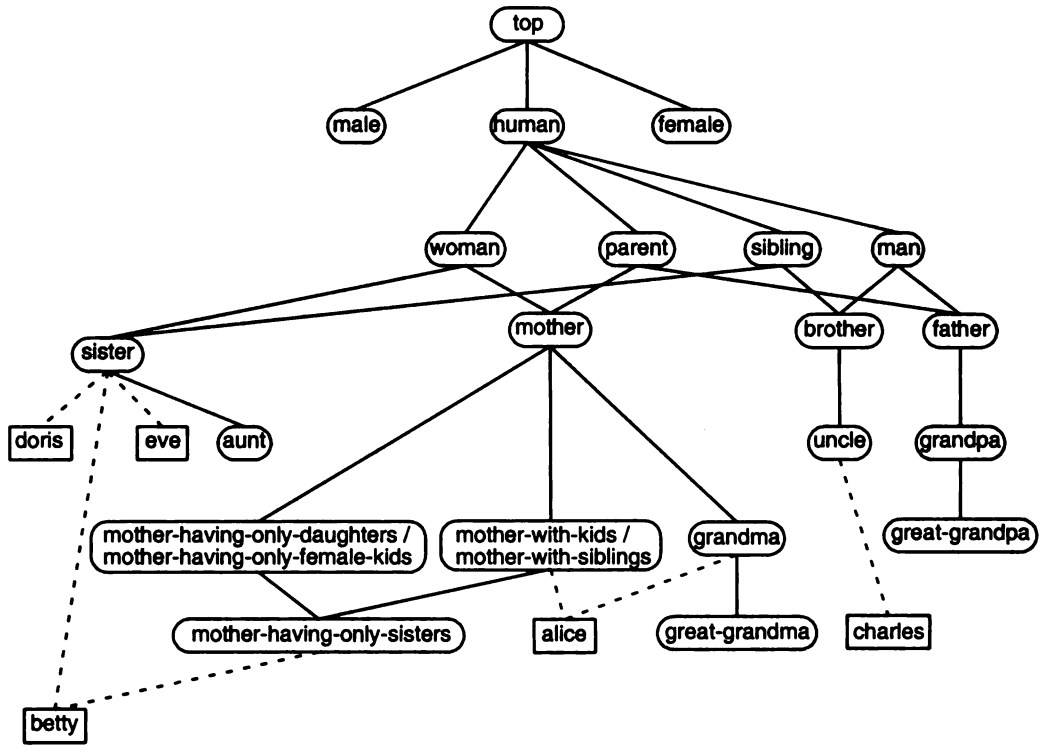


Figure 1: Concept hierarchy of the TBox *family* augmented with the individuals from the ABox *smith_family*. Ovals represent atomic concepts, rectangles denote ABox individuals, solid lines show the direct subsumption relationship, and dashed lines the instance membership of the individuals for their direct types.

The next axioms guarantee the disjointness between the concepts female, male, and human.

$\mathsf{female} \sqsubseteq \neg(\mathsf{human} \sqcup \mathsf{male})$

$\mathsf{male} \sqsubseteq \neg(\mathsf{human} \sqcup \mathsf{female})$

$\mathsf{human} \sqsubseteq \neg(\mathsf{female} \sqcup \mathsf{male})$

After these preliminaries we start with axioms expressing basic knowledge about family members. We use $\mathsf{C} \doteq \mathsf{D}$ as an abbreviation for $\mathsf{C} \sqsubseteq \mathsf{D}$ and $\mathsf{D} \sqsubseteq \mathsf{C}$.

$\mathsf{human} \sqsubseteq \exists_{\geq 1}\, \mathsf{has\_gender}$

$\mathsf{woman} \doteq \mathsf{human} \sqcap \forall\, \mathsf{has\_gender} \,.\, \mathsf{female}$

$\mathsf{man} \doteq \mathsf{human} \sqcap \forall\, \mathsf{has\_gender} \,.\, \mathsf{male}$

$\mathsf{parent} \doteq \exists_{\geq 1}\, \mathsf{has\_child}$

$\mathsf{mother} \doteq \mathsf{woman} \sqcap \mathsf{parent}$

$\mathsf{father} \doteq \mathsf{man} \sqcap \mathsf{parent}$

The next axioms describe some aspects of relatives of a family. Note the inferred equivalences between the concept pairs "mother_with_..." and "mother_having_..." as shown in Figure 1.

$\mathsf{mother\_having\_only\_female\_kids} \doteq$
$\mathsf{mother} \sqcap \forall\, \mathsf{has\_child} \,.\, \forall\, \mathsf{has\_gender} \,.\, \mathsf{female}$

$\mathsf{mother\_having\_only\_daughters} \doteq$
$\mathsf{mother} \sqcap \exists_{\geq 1}\, \mathsf{has\_child} \sqcap \forall\, \mathsf{has\_child} \,.\, \mathsf{woman}$

$\mathsf{mother\_with\_kids} \doteq \mathsf{mother} \sqcap \exists_{\geq 2}\, \mathsf{has\_child}$

$\mathsf{grandpa} \doteq \mathsf{man} \sqcap \exists\, \mathsf{has\_child} \,.\, \mathsf{parent}$

$\mathsf{great\_grandpa} \doteq \mathsf{man} \sqcap \exists\, \mathsf{has\_child} \,.\, (\exists\, \mathsf{has\_child} \,.\, \mathsf{parent})$

$\mathsf{grandma} \doteq \mathsf{woman} \sqcap \exists\, \mathsf{has\_child} \,.\, \mathsf{parent}$

$\mathsf{great\_grandma} \doteq$
$\mathsf{woman} \sqcap \exists\, \mathsf{has\_child} \,.\, (\exists\, \mathsf{has\_child} \,.\, \mathsf{parent})$

$\mathsf{aunt} \doteq \mathsf{woman} \sqcap \exists\, \mathsf{has\_sibling} \,.\, \mathsf{parent}$

$\mathsf{uncle} \doteq \mathsf{man} \sqcap \exists\, \mathsf{has\_sibling} \,.\, \mathsf{parent}$

$\mathsf{sibling} \doteq \mathsf{sister} \sqcup \mathsf{brother}$

$\mathsf{sister} \doteq \mathsf{woman} \sqcap \exists_{\geq 1}\, \mathsf{has\_sibling}$

$\mathsf{brother} \doteq \mathsf{man} \sqcap \exists_{\geq 1}\, \mathsf{has\_sibling}$

$\mathsf{mother\_with\_siblings} \doteq \mathsf{mother} \sqcap \forall\, \mathsf{has\_child} \,.\, \mathsf{sibling}$

There still exists no formal relationship between the notions "having kids" and "having siblings." This

is expressed by the next two axioms. The last axiom defines a concept mother_having_only_sisters which has the other specific "mother_..." concepts as parents (see Figure 1).

$\exists_{\geq 2}$ has_child $\sqsubseteq \forall$ has_child . sibling

$\exists$ has_child . sibling $\sqsubseteq \exists_{\geq 2}$ has_child

mother_having_only_sisters $\doteq$
  mother $\sqcap \forall$ has_child . (sister $\sqcap \forall$ has_sibling . sister)

Using the TBox *family*, the ABox *smith_family* is specified. It consists of several assertions about the individuals alice, betty, charles, doris, and eve. The individual alice is the mother of her two children betty and charles.

alice : woman $\sqcap \exists_{\leq 2}$ has_child
(alice, betty) : has_child
(alice, charles) : has_child

The individual betty is the sibling of charles and the mother of doris and eve, who are the only siblings of each other. The individual charles is the only brother of betty.

betty : woman $\sqcap \exists_{\leq 2}$ has_child $\sqcap \exists_{\leq 1}$ has_sibling
(betty, doris) : has_child
(betty, eve) : has_child
(betty, charles) : has_sibling
charles : brother $\sqcap \exists_{\leq 1}$ has_sibling
(charles, betty) : has_sibling
doris : $\exists_{\leq 1}$ has_sibling
eve : $\exists_{\leq 1}$ has_sibling
(doris, eve) : has_sister
(eve, doris) : has_sister

Figure 1 also shows the inferred *direct types* of the individuals in ABox *smith_family*. alice has as direct types {mother_with_siblings, grandma}, betty has {mother_having_only_sisters, sister}, charles has {uncle}, and doris and eve have {sister}. These inferences demonstrate the expressiveness of $\mathcal{ALCNH}_{R^+}$. The ABox *smith_family* contains only minimal knowledge about the individuals and their relationships.

4 A Tableaux Calculus for $\mathcal{ALCNH}_{R^+}$

In the following we devise a *tableaux* algorithm to decide the consistency of $\mathcal{ALCNH}_{R^+}$ ABoxes. The algorithm is characterized by a set of tableaux or *completion* rules and by a particular *completion strategy* ensuring a specific order for applying the completion rules to assertional axioms of an ABox. The strategy is essential to guarantee the completeness of the ABox consistency algorithm. First, we have to introduce new assertional axioms needed to define the augmentation of an ABox.

Definition 7 (Additional ABox Assertions) Let C be a concept term, the individual names a, b $\in O$, and $x \notin O$, then the following expressions are also assertional axioms:

- $\forall x . x$:C *(universal concept assertion)*,[1]
- a $\neq$ b *(inequality assertion)*.

An interpretation $\mathcal{I}$ satisfies an assertional axiom $\forall x . x$:C iff $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and a $\neq$ b iff $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.

Given the new ABox assertions we define for any concept term its negation normal form that is needed to introduce the notion of an augmented ABox.

Definition 8 (Negation Normal Form) The same naming conventions as in Definition 3 are assumed. The negation normal form is defined by applying the following transformations in such a way that a negation sign may occur only in front of concept names. This transformation is possible in linear time:

- $\neg\top \equiv \bot$,
- $\neg\bot \equiv \top$,
- $\neg(C \sqcap D) \equiv \neg C \sqcup \neg D$,
- $\neg(C \sqcup D) \equiv \neg C \sqcap \neg D$,
- $\neg\forall R . C \equiv \exists R . \neg C$,
- $\neg\exists R . C \equiv \forall R . \neg C$,
- $\neg\exists_{\leq m} S \equiv \exists_{\geq m+1} S$,
- $\neg\exists_{\geq m} S \equiv \exists_{\leq m-1} S$.

Definition 9 (Augmented ABox) For an initial ABox $\mathcal{A}$ w.r.t a TBox $\mathcal{T}$ we define its *augmented* ABox $\mathcal{A}'$ by applying the following rules to $\mathcal{A}$. For every GCI $C \sqsubseteq D$ in $\mathcal{T}$ the assertion $\forall x . x$:$(\neg C \sqcup D)$ is added to $\mathcal{A}'$. Every concept term occurring in $\mathcal{A}$ is transformed into its negation normal form. Let $O_O := \{a_1, \ldots, a_n\}$ be the set of old individual names mentioned in $\mathcal{A}$, then the set of inequality assertions $\{a_i \neq a_j \mid a_i, a_j \in O_O, i, j \in 1..n, i \neq j\}$ is added to $\mathcal{A}$. From this point on, if we refer to an initial ABox $\mathcal{A}$ we always mean its augmented ABox.

The tableaux rules also require the notion of *blocking* their applicability. This is based on so-called concept sets.

[1] $\forall x . x$:C should be read as $\forall x . (x$:C).

Definition 10 (Concept Set, $\mathcal{A}$-blocked)
Given an ABox $\mathcal{A}$ and an individual a occurring in $\mathcal{A}$, we define the *concept set* of a as $\sigma(\mathcal{A}, a) := \{\top\} \cup \{C \mid a{:}C \in \mathcal{A}\}$. We define two individuals as *$\mathcal{A}$-equivalent*, written $a \equiv_{\mathcal{A}} b$, if their concept sets are equal, i.e. $\sigma(\mathcal{A}, a) = \sigma(\mathcal{A}, b)$. We say that an individual b is *$\mathcal{A}$-blocked*[2] by a, written $a \succcurlyeq_{\mathcal{A}} b$, if $\sigma(\mathcal{A}, a) \supseteq \sigma(\mathcal{A}, b)$.

4.1 Completion Rules

We are now ready to define the *completion rules* that are intended to generate a so-called completion of an ABox (see also below).

Definition 11 (Completion Rules)

R⊓ The conjunction rule.
if 1. $a{:}C \sqcap D \in \mathcal{A}$, and
2. $\{a{:}C,\ a{:}D\} \not\subseteq \mathcal{A}$
then $\mathcal{A}' = \mathcal{A} \cup \{a{:}C,\ a{:}D\}$

R⊔ The disjunction rule.
if 1. $a{:}C \sqcup D \in \mathcal{A}$, and
2. $\{a{:}C,\ a{:}D\} \cap \mathcal{A} = \emptyset$
then $\mathcal{A}' = \mathcal{A} \cup \{a{:}C\}$ **or** $\mathcal{A}' = \mathcal{A} \cup \{a{:}D\}$

R∀C The role value restriction rule.
if 1. $a{:}\forall R.C \in \mathcal{A}$, and
2. $\exists b \in O, S \in R^{\downarrow} : (a,b){:}S \in \mathcal{A}$, and
3. $b{:}C \notin \mathcal{A}$
then $\mathcal{A}' = \mathcal{A} \cup \{b{:}C\}$

R∀$_+$C The transitive role value restriction rule.
if 1. $a{:}\forall R.C \in \mathcal{A}$, and
2. $\exists b \in O, T \in R^{\downarrow}, T \in \mathcal{T}, S \in T^{\downarrow}$:
$(a,b){:}S \in \mathcal{A}$, and
3. $b{:}\forall T.C \notin \mathcal{A}$
then $\mathcal{A}' = \mathcal{A} \cup \{b{:}\forall T.C\}$

R∀$_x$ The universal concept restriction rule.
if 1. $\forall x.x{:}C \in \mathcal{A}$, and
2. $\exists a \in O$: a mentioned in $\mathcal{A}$, and
3. $a{:}C \notin \mathcal{A}$
then $\mathcal{A}' = \mathcal{A} \cup \{a{:}C\}$

R∃C The role exists restriction rule.
if 1. $a{:}\exists R.C \in \mathcal{A}$, and
2. $a \in O_N \Rightarrow$
$(\neg\exists c \in O_N : c$ mentioned in $\mathcal{A}, c \succcurlyeq_{\mathcal{A}} a)$, and
3. $\neg\exists b \in O, S \in R^{\downarrow} : \{(a,b){:}S,\ b{:}C\} \subseteq \mathcal{A}$
then $\mathcal{A}' = \mathcal{A} \cup \{(a,b){:}R,\ b{:}C\}$ where $b \in O_N$ is not used in $\mathcal{A}$

R∃$_{\geq n}$ The number restriction exists rule.
if 1. $a{:}\exists_{\geq n} R \in \mathcal{A}$, and
2. $a \in O_N \Rightarrow$
$(\neg\exists c \in O_N : c$ mentioned in $\mathcal{A}, c \succcurlyeq_{\mathcal{A}} a)$, and
3. $\neg\exists b_1, \ldots, b_n \in O, S_1, \ldots, S_n \in R^{\downarrow}$:
$\{(a,b_k){:}S_k \mid k \in 1..n\} \cup$
$\{b_i \neq b_j \mid i,j \in 1..n, i \neq j\} \subseteq \mathcal{A}$
then $\mathcal{A}' = \mathcal{A} \cup \{(a,b_k){:}R \mid k \in 1..n\} \cup$
$\{b_i \neq b_j \mid i,j \in 1..n, i \neq j\}$
where $b_1, \ldots, b_n \in O_N$ are not used in $\mathcal{A}$

R∃$_{\leq n}$ The number restriction merge rule.
if 1. $a{:}\exists_{\leq n} R \in \mathcal{A}$, and
2. $\exists b_1, \ldots, b_m \in O, S_1, \ldots, S_m \in R^{\downarrow}$:
$\{(a,b_1){:}S_1, \ldots, (a,b_m){:}S_m\} \subseteq \mathcal{A}$
with $m > n$, and
3. $\exists b_i, b_j \in \{b_1, \ldots, b_m\} : i \neq j,\ b_i \neq b_j \notin \mathcal{A}$
then $\mathcal{A}' = \mathcal{A}[b_i/b_j]$, i.e. replace every occurrence of b_i in $\mathcal{A}$ by b_j

We call the rules R⊔ and R∃$_{\leq n}$ *nondeterministic* rules since they can be applied in different ways to the same ABox. The remaining rules are called *deterministic* rules. Moreover, we call the rules R∃C and R∃$_{\geq n}$ *generating* rules since they are the only rules that introduce new individuals in an ABox.

The increase of expressiveness in $\mathcal{ALCNH}_{R^+}$ gained by supporting ABox reasoning is reflected in tableaux rules that are more complex than in comparable approaches for concept consistency. The universal concept restriction rule takes care of GCIs and usually causes additional complexity by adding disjunctions to an ABox. The generating rules have a more complex premise since they may test only for a blocking situation if they are applied to new individuals, i.e. a blocking situation can never occur for old individuals. The necessity of this additional precondition is illustrated by the following example. We define a concept D where R is a transitive superrole of S.

$$D \doteq C \sqcap \exists S.C \sqcap \exists_{\leq 1} S \sqcap \forall R.\exists S.C$$
$$\mathcal{A} := \{(i,j) : S,\ (j,k) : S,\ i : D,\ j : D,\ k : \neg C\}$$

Then, we define an ABox $\mathcal{A}$ which is obviously unsatisfiable due to a clash for the individual k with $C \sqcap \neg C$. However, if blocking were allowed for old individuals, the role exists restriction rule would not create a S-successor with qualification C for the individual j. As a consequence, the number restriction merge rule would never merge this successor with the individual k which results in the unsatisfiability of $\mathcal{A}$.

[2] We may omit the reference to $\mathcal{A}$ by speaking of *blocked* if the context is obvious.

Proposition 12 (Invariance) Let $\mathcal{A}$ and $\mathcal{A}'$ be ABoxes. Then:

1. If $\mathcal{A}'$ is derived from $\mathcal{A}$ by applying a deterministic rule, then $\mathcal{A}$ is satisfiable iff $\mathcal{A}'$ is satisfiable.
2. If $\mathcal{A}'$ is derived from $\mathcal{A}$ by applying a nondeterministic rule, then $\mathcal{A}$ is satisfiable if $\mathcal{A}'$ is satisfiable. Conversely, if $\mathcal{A}$ is satisfiable and a nondeterministic rule is applicable to $\mathcal{A}$, then it can be applied in such a way that it yields a satisfiable ABox $\mathcal{A}'$.

Proof. **1.** "$\Leftarrow$" Due to the structure of the deterministic rules one can immediately verify that $\mathcal{A}$ is a subset of $\mathcal{A}'$. Therefore, $\mathcal{A}$ is satisfiable if $\mathcal{A}'$ is satisfiable.

"$\Rightarrow$" In order to show that $\mathcal{A}'$ is satisfiable after applying a deterministic rule to the satisfiable ABox $\mathcal{A}$, we examine each applicable rule separately. We assume that $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ satisfies $\mathcal{A}$.

If the conjunction rule is applied to $a:C \sqcap D \in \mathcal{A}$, then we get a new Abox $\mathcal{A}' = \mathcal{A} \cup \{a:C,\ a:D\}$. Since $\mathcal{I}$ satisfies $a:C \sqcap D$, $\mathcal{I}$ satisfies $a:C$ and $a:D$ and therefore $\mathcal{A}'$.

If the role value restriction rule is applied to $a:\forall R.C \in \mathcal{A}$, then there must be a role assertion $(a,b):S \in \mathcal{A}$ with $S \in R^{\downarrow}$ such that $\mathcal{A}' = \mathcal{A} \cup \{b:C\}$. Since $\mathcal{I}$ satisfies $\mathcal{A}$, it holds that $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in S^{\mathcal{I}} \subseteq R^{\mathcal{I}}$. Since $\mathcal{I}$ satisfies $a:\forall R.C$, it holds that $b^{\mathcal{I}} \in C^{\mathcal{I}}$. Thus, $\mathcal{I}$ satisfies $b:C$ and therefore $\mathcal{A}'$.

If the transitive role value restriction rule is applied to $a:\forall R.C \in \mathcal{A}$, there must be an assertion $(a,b):S \in \mathcal{A}$ with $S \in T^{\downarrow} \subseteq R^{\downarrow}$, $T \in \mathcal{T}$ such that we get $\mathcal{A}' = \mathcal{A} \cup \{b:\forall T.C\}$. Since $\mathcal{I}$ satisfies $\mathcal{A}$, we have $a^{\mathcal{I}} \in (\forall R.C)^{\mathcal{I}}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in S^{\mathcal{I}} \subseteq T^{\mathcal{I}} \subseteq R^{\mathcal{I}}$. Since $\mathcal{I}$ satisfies $a:\forall T.C$ and $T \in \mathcal{T}, T \in R^{\downarrow}$, it holds that $b^{\mathcal{I}} \in (\forall T.C)^{\mathcal{I}}$ unless there exists a successor c of b such that $(b,c):S' \in \mathcal{A}$, $(b^{\mathcal{I}}, c^{\mathcal{I}}) \in S'^{\mathcal{I}} \subseteq T^{\mathcal{I}}$ and $c^{\mathcal{I}} \notin C^{\mathcal{I}}$. It follows from $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in T^{\mathcal{I}}$, $(b^{\mathcal{I}}, c^{\mathcal{I}}) \in T^{\mathcal{I}}$, and $T \in \mathcal{T}$ that $(a^{\mathcal{I}}, c^{\mathcal{I}}) \in T^{\mathcal{I}} \subseteq R^{\mathcal{I}}$ and $a^{\mathcal{I}} \notin (\forall R.C)^{\mathcal{I}}$ in contradiction to the assumption. Thus, $\mathcal{I}$ satisfies $b:\forall T.C$ and therefore $\mathcal{A}'$.

If the universal concept restriction rule is applied to an individual a in $\mathcal{A}$ because of $\forall x.x:C \in \mathcal{A}$, then $\mathcal{A}' = \mathcal{A} \cup \{a:C\}$. Since $\mathcal{I}$ satisfies $\mathcal{A}$, it holds that $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$. Thus, it holds that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $\mathcal{I}$ satisfies $\mathcal{A}'$.

If the role exists restriction rule is applied to $a:\exists R.C \in \mathcal{A}$, then we get the ABox $\mathcal{A}' = \mathcal{A} \cup \{(a,b):R,\ b:C\}$. Since $\mathcal{I}$ satisfies $\mathcal{A}$, there exists a $y \in \Delta^{\mathcal{I}}$ such that $(a^{\mathcal{I}}, y) \in R^{\mathcal{I}}$ and $y \in C^{\mathcal{I}}$. We define the interpretation function $\cdot^{\mathcal{I}'}$ such that $b^{\mathcal{I}'} := y$ and $x^{\mathcal{I}'} := x^{\mathcal{I}}$ for $x \neq b$. It is easy to show that $\mathcal{I}' = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}'})$ satisfies $\mathcal{A}'$.

If the number restriction exists rule is applied to $a:\exists_{\geq n} R \in \mathcal{A}$, then we get $\mathcal{A}' = \mathcal{A} \cup \{(a,b_k):R \mid k \in 1..n\} \cup \{b_i \neq b_j \mid i,j \in 1..n, i \neq j\}$. Since $\mathcal{I}$ satisfies $\mathcal{A}$, there must exist n distinct individuals $y_i \in \Delta^{\mathcal{I}}$, $i \in 1..n$ such that $(a^{\mathcal{I}}, y_i) \in R^{\mathcal{I}}$. We define the interpretation function $\cdot^{\mathcal{I}'}$ such that $b_i^{\mathcal{I}'} := y_i$ and $x^{\mathcal{I}'} := x^{\mathcal{I}}$ for $x \notin \{b_1, \ldots, b_n\}$. It is easy to show that $\mathcal{I}' = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}'})$ satisfies $\mathcal{A}'$.

2. "$\Leftarrow$" Assume that $\mathcal{A}'$ is satisfied by $\mathcal{I}' = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}'})$. We show that $\mathcal{A}$ is also satisfiable by examining the nondeterministic rules.

If $\mathcal{A}'$ is obtained from $\mathcal{A}$ by applying the disjunction rule, then $\mathcal{A}$ is a subset of $\mathcal{A}'$ and therefore satisfied by $\mathcal{I}'$.

If $\mathcal{A}'$ is obtained from $\mathcal{A}$ by applying the number restriction merge rule to $a:\exists_{\leq n} R \in \mathcal{A}$, then there exist b_i, b_j in $\mathcal{A}$ such that $\mathcal{A}' = \mathcal{A}[b_i/b_j]$. We define the interpretation function $\cdot^{\mathcal{I}}$ such that $b_i^{\mathcal{I}} := b_j^{\mathcal{I}'}$ and $x^{\mathcal{I}} := x^{\mathcal{I}'}$ for every $x \neq b_i$. Obviously $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ satisfies $\mathcal{A}$.

"$\Rightarrow$" We suppose that $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ satisfies $\mathcal{A}$ and a nondeterministic rule is applicable to an individual a in $\mathcal{A}$.

If the disjunction rule is applicable to $a:C \sqcup D \in \mathcal{A}$ and $\mathcal{A}$ is satisfiable, it holds $a^{\mathcal{I}} \in (C \sqcup D)^{\mathcal{I}}$. It follows that either $a^{\mathcal{I}} \in C^{\mathcal{I}}$ or $a^{\mathcal{I}} \in D^{\mathcal{I}}$ (or both). Hence, the disjunction rule can be applied in a way that $\mathcal{I}$ also satisfies the ABox $\mathcal{A}'$.

If the number restriction merge rule is applicable to $a:\exists_{\leq n} R \in \mathcal{A}$ and $\mathcal{A}$ is satisfiable, it holds $a^{\mathcal{I}} \in (\exists_{\leq n} R)^{\mathcal{I}}$ and $\|\{b \mid (a,b) \in R^{\mathcal{I}}\}\| \leq n$. However, it also holds $\|\{b \mid (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}\}\| > m$ with $m \geq n$.[3] Thus, we can conclude by the Pigeonhole Principle (e.g. see [13, page 26]) that there exist at least two R-successors b_i, b_j of a such that $b_i^{\mathcal{I}} = b_j^{\mathcal{I}}$. Since $\mathcal{I}$ satisfies $\mathcal{A}$, we have $b_i \neq b_j \notin \mathcal{A}$ and at least one of the two individuals must be a new individual. Let us assume that $b_i \in O_N$ and $b_i = b_j$, then $\mathcal{I}$ obviously satisfies $\mathcal{A}[b_i/b_j]$. □

Given an initial ABox $\mathcal{A}$, more than one rule might be applicable to $\mathcal{A}$. This is controlled by a completion strategy in accordance to an ordering for new individuals.

[3] Without loss of generality we only need to consider the case that $m = n + 1$.

Definition 13 (Individual Ordering) We define an *individual ordering* '$\prec$' for new individuals (elements of O_N) occurring in an ABox $\mathcal{A}$. If $b \in O_N$ is introduced in $\mathcal{A}$, then $a \prec b$ for all new individuals a already present in $\mathcal{A}$.

Definition 14 (Completion Strategy) We define a *completion strategy* that must observe the following restrictions.

- Meta rules:
  - Apply a rule to an individual $b \in O_N$ only if no rule is applicable to an individual $a \in O_O$.
  - Apply a rule to an individual $b \in O_N$ only if no rule is applicable to another individual $a \in O_N$ such that $a \prec b$.
- The completion rules are always applied in the following order. A step is skipped in case the corresponding set of applicable rules is empty.
  1. Apply all nongenerating rules (R$\sqcap$, R$\sqcup$, R$\forall$C, R$\forall_+$C, R$\forall_x$, R$\exists_{\leq n}$) as long as possible.
  2. Apply a generating rule (R$\exists$C, R$\exists_{\geq n}$) and restart with step 1 as long as possible.

In the following we always assume that rules are applied in accordance to this strategy. It ensures that the rules are applied to new individuals w.r.t. the ordering '$\prec$'.

Definition 15 (Clash Triggers) We assume the same naming conventions as used above. An ABox $\mathcal{A}$ is called *contradictory* if one of the following *clash triggers* is applicable. If none of the clash triggers is applicable to $\mathcal{A}$, then $\mathcal{A}$ is called *clash-free*.

- *Primitive clash*:
  $a:\bot \in \mathcal{A}$ or $\{a:C, a:\neg C\} \subseteq \mathcal{A}$, where C is a concept name.
- *Number restriction merging clash*:
  $\{\exists_{\leq n} R\} \cup \{(a, b_k):S_k \mid S_k \in R^{\downarrow}, k \in 1..m\} \cup \{b_i \neq b_j \mid i,j \in 1..m, i \neq j\} \subseteq \mathcal{A}$ with $m > n$

A clash-free ABox $\mathcal{A}$ is called *complete* if no completion rule is applicable to $\mathcal{A}$. A complete ABox $\mathcal{A}'$ derived from an ABox $\mathcal{A}$ is also called a *completion* of $\mathcal{A}$. Any ABox containing a clash is obviously unsatisfiable. The purpose of the calculus is to generate a completion for an initial ABox $\mathcal{A}$ that proves the satisfiability of $\mathcal{A}$ or its unsatisfiability if no completion can be found. In the following we have to show that a model can be constructed for any complete ABox.

4.2 Decidability of the ABox Consistency Problem

The following lemma proves that whenever a generating rule has been applied to an individual a, the concept set $\sigma(\cdot, a)$ of a does not change in succeeding ABoxes.

Lemma 16 (Stability) Let $\mathcal{A}$ be an ABox and $a \in O_N$ be in $\mathcal{A}$. Let a generating rule be applicable to a according to the completion strategy. Let $\mathcal{A}'$ be any ABox derivable from $\mathcal{A}$ by any (possibly empty) sequence of rule applications. Then:

1. No rule is applicable in $\mathcal{A}'$ to an individual $b \in O_N$ with $b \prec a$
2. $\sigma(\mathcal{A}, a) = \sigma(\mathcal{A}', a)$, i.e. the concept set of a remains unchanged in $\mathcal{A}'$.
3. If $b \in O_N$ is in $\mathcal{A}$ with $b \prec a$ then b is an individual in $\mathcal{A}'$, i.e. the individual b is not substituted by another individual.

Proof. **1.** By contradiction: Suppose $\mathcal{A} = \mathcal{A}_0 \rightarrow_* \cdots \rightarrow_* \mathcal{A}_n = \mathcal{A}'$, where $*$ is element of the completion rules and a rule is applicable to an individual b with $b \prec a$ in $\mathcal{A}'$. Then there has to exist a minimal i with $i \in 1..n$ such that this rule is also applicable in $\mathcal{A}_i$. If a rule is applicable to a in $\mathcal{A}$ then no rule is applicable to b in $\mathcal{A}$ due to our strategy. So no rule is applicable to any individual c such that $c \prec a$ in $\mathcal{A}_0, \ldots, \mathcal{A}_{i-1}$. It follows that from $\mathcal{A}_{i-1}$ to $\mathcal{A}_i$ a rule is applied to a or to a d such that $a \prec d$. Using an exhaustive case analysis of all rules we can show that no new assertion of the form $b:C$ or $(b,e):R$ can be added to $\mathcal{A}_{i-1}$. Therefore, no rule is applicable to b in $\mathcal{A}_i$. This is a contradiction to our assumption.

2. By contradiction: Suppose $\sigma(\mathcal{A}, a) \neq \sigma(\mathcal{A}', a)$. Let b be the direct predecessor of a with $b \prec a$. A rule must have been applied to a and not to b because of point 1. Due to our strategy only generating rules are applicable to a that cannot add new elements to $\sigma(\cdot, a)$. This is an obvious contradiction.

3. This follows from point 1 and the completion strategy. □

Definition 17 (Blocking Individual) Let $\mathcal{A}$ be an ABox and $a, b \in O_N$ be individuals in $\mathcal{A}$. We call a the blocking individual of b if the following conditions hold:

- $a \succcurlyeq_{\mathcal{A}} b$,
- $a \prec b$,
- $\neg\exists c$ in $\mathcal{A}: c \in O_N, c \prec a, c \succcurlyeq_{\mathcal{A}} b$.

The next lemma guarantees the uniqueness of a blocking individual for a blocked individual. This is a precondition for defining a particular interpretation from $\mathcal{A}$.

Lemma 18 Let $\mathcal{A}'$ be an ABox and a be a new individual in $\mathcal{A}'$. If a is blocked then

1. a has no direct successor and
2. a has exactly one blocking individual.

Proof. **1.** By contradiction: Suppose that a is blocked in $\mathcal{A}'$ and $(a,b):R \in \mathcal{A}'$. There must exist an ancestor ABox $\mathcal{A}$ where a generating rule has been applied to a in $\mathcal{A}$. It follows from the definition of the generating rules that for every new individual c with $c \prec a$ in $\mathcal{A}$ we had $\sigma(\mathcal{A},c) \not\supseteq \sigma(\mathcal{A},a)$. Since $\mathcal{A}'$ has been derived from $\mathcal{A}$ we can use Lemma 16 and conclude that for every new individual c with $c \prec a$ in $\mathcal{A}'$ we also have $\sigma(\mathcal{A}',c) \not\supseteq \sigma(\mathcal{A}',a)$. Thus there cannot exist a blocking individual c for a in $\mathcal{A}'$. This is a contradiction to our hypothesis.

2. This follows directly from condition 3 in Definition 17. □

Definition 19 Let A be an ABox. We define the *canonical interpretation* $\mathcal{I}_\mathcal{A} = (\Delta^{\mathcal{I}_\mathcal{A}}, \cdot^{\mathcal{I}_\mathcal{A}})$ as follows:

1. $\Delta^{\mathcal{I}_\mathcal{A}} := \{a \mid a \text{ is an individual in } \mathcal{A}\}$
2. $a^{\mathcal{I}_\mathcal{A}} := a$ iff a is mentioned in $\mathcal{A}$
3. $a \in A^{\mathcal{I}_\mathcal{A}}$ iff $a:A \in \mathcal{A}$
4. $(a,b) \in R^{\mathcal{I}_\mathcal{A}}$ iff
   (a) $(a,b):S \in \mathcal{A}$ for a role $S \in R^{\downarrow}$ *or*
   (b) $\exists c_1,\ldots,c_{n-1}$ in $\mathcal{A}$: $(a,c_1):S_1$, $(c_1,c_2):S_2$, $\ldots$, $(c_{n-1},b):S_n \in \mathcal{A}$, $n > 1$, $S_i \in R^{\downarrow}$ for $i \in 1..n$ and $R \in T$, *or*
   (c) $\exists c$ in $\mathcal{A}$, $c \in O_N$, c is a blocking individual for a, and $(c,b):S \in \mathcal{A}$, for a role $S \in R^{\downarrow}$, *or*
   (d) $\exists c$ in $\mathcal{A}$, $c \in O_N$, c is a blocking individual for a, and $(c,b_1):S_1 \in \mathcal{A}$, and $\exists b_2,\ldots,b_{n-1}$ in $\mathcal{A}$: $(b_1,b_2):S_2$, $\ldots$, $(b_{n-1},b):S_n \in \mathcal{A}$, $n > 1$, $S_i \in R^{\downarrow}$ for $i \in 1..n$ and $R \in T$.

Theorem 20 (Soundness) Let $\mathcal{A}$ be a complete ABox, then $\mathcal{A}$ is satisfiable.

Proof. Let $\mathcal{I}_\mathcal{A} = (\Delta^{\mathcal{I}_\mathcal{A}}, \cdot^{\mathcal{I}_\mathcal{A}})$ be the canonical interpretation for the ABox $\mathcal{A}$. In the following we prove that $\mathcal{I}_\mathcal{A}$ satisfies every assertion in $\mathcal{A}$.

For any $(a,b):R \in \mathcal{A}$ or $a \neq b \in \mathcal{A}$, $\mathcal{I}_\mathcal{A}$ satisfies them by definition. Next we consider assertions of the form $a:C$. We show by induction on the structure of C that $a \in C^{\mathcal{I}_\mathcal{A}}$.

If C is a concept name, then $a \in C^{\mathcal{I}_\mathcal{A}}$ by definition of $\mathcal{I}_\mathcal{A}$. If $C = \top$, then obviously $a \in \top^{\mathcal{I}_\mathcal{A}}$. The case $C = \bot$ cannot occur since $\mathcal{A}$ is clash-free.

If $C = \neg D$, then D is a concept name since all concepts are in negation normal form (see Definition 9). $\mathcal{A}$ is clash-free and cannot contain $a:D$. Thus, $a \notin D^{\mathcal{I}_\mathcal{A}}$, i.e. $a \in \Delta^{\mathcal{I}_\mathcal{A}} \setminus D^{\mathcal{I}_\mathcal{A}}$. Hence $a \in (\neg D)^{\mathcal{I}_\mathcal{A}}$.

If $C = C_1 \sqcap C_2$ then (since $\mathcal{A}$ is complete) $a:C_1 \in \mathcal{A}$ and $a:C_2 \in \mathcal{A}$. By induction hypothesis, $a \in C_1^{\mathcal{I}_\mathcal{A}}$ and $a \in C_2^{\mathcal{I}_\mathcal{A}}$. Hence $a \in (C_1 \sqcap C_2)^{\mathcal{I}_\mathcal{A}}$.

If $C = C_1 \sqcup C_2$ then (since $\mathcal{A}$ is complete) either $a:C_1 \in \mathcal{A}$ or $a:C_2 \in \mathcal{A}$. By induction hypothesis, $a \in C_1^{\mathcal{I}_\mathcal{A}}$ or $a \in C_2^{\mathcal{I}_\mathcal{A}}$. Hence $a \in (C_1 \sqcup C_2)^{\mathcal{I}_\mathcal{A}}$.

If $C = \forall R.D$, then we have to show that for all b with $(a,b) \in R^{\mathcal{I}_\mathcal{A}}$ it holds that $b \in D^{\mathcal{I}_\mathcal{A}}$. If $(a,b) \in R^{\mathcal{I}_\mathcal{A}}$, then according to Definition 19 the following cases can occur: (4a) b is a direct S-successor of a for a role $S \in R^{\downarrow}$ with $S^{\mathcal{I}_\mathcal{A}} \subseteq R^{\mathcal{I}_\mathcal{A}}$; then we have $b:D \in \mathcal{A}$ since $\mathcal{A}$ is complete and by induction hypothesis $b \in D^{\mathcal{I}_\mathcal{A}}$. (4b) b is a R-successor of a via a subrole chain of S_i's with $S_i^{\mathcal{I}_\mathcal{A}} \subseteq R^{\mathcal{I}_\mathcal{A}}$, $R \in T$; then we have $c_{n-1}:\forall R.D \in \mathcal{A}$ and $b:D \in \mathcal{A}$ since $\mathcal{A}$ is complete and by induction hypothesis we have $b \in D^{\mathcal{I}_\mathcal{A}}$. (4c) There has to exist a blocking individual c such that $c:\forall R.D \in \mathcal{A}$ and $(c,b):S \in \mathcal{A}$ for a role $S \in R^{\downarrow}$ and because $\mathcal{A}$ is complete we have $b:D \in \mathcal{A}$ and again by induction hypothesis it holds $b \in D^{\mathcal{I}_\mathcal{A}}$. (4d) This case combines the cases (4b-c) because the individual b is reachable from the blocking individual c via a chain of subroles of the transitive role R. It can be proven analogously.

If $C = \exists R.D$, then we have to show that there exists an individual $b \in \Delta^{\mathcal{I}_\mathcal{A}}$ with $(a,b) \in R^{\mathcal{I}_\mathcal{A}}$ and $b \in D^{\mathcal{I}_\mathcal{A}}$. Since ABox $\mathcal{A}$ is complete, we have either $(a,b):R \in \mathcal{A}$ and $b:D \in \mathcal{A}$ or a is blocked by an individual c and $(c,b):R \in \mathcal{A}$. In the first case we have $(a,b) \in R^{\mathcal{I}_\mathcal{A}}$ and $b \in D^{\mathcal{I}_\mathcal{A}}$ by induction hypothesis and the definition of $\mathcal{I}_\mathcal{A}$. In the second case there exists the blocking individual c with $c:\exists R.D \in \mathcal{A}$. By definition c cannot be blocked and by hypothesis $\mathcal{A}$ is complete. So we have an individual b with $(c,b):R \in \mathcal{A}$ and $b:D \in \mathcal{A}$. By induction hypothesis we have $b \in D^{\mathcal{I}_\mathcal{A}}$ and by the definition of $\mathcal{I}_\mathcal{A}$ (case 4c) we have $(a,b) \in R^{\mathcal{I}_\mathcal{A}}$.

If $C = \exists_{\geq n} R$, we prove the hypothesis by contradiction. We assume that $a \notin (\exists_{\geq n} R)^{\mathcal{I}_\mathcal{A}}$. Then there exist at most m $(0 \leq m < n)$ distinct R-successors of a. Two cases can occur: (1) the individual a is not blocked in $\mathcal{I}_\mathcal{A}$. Then we have less than n R-successors of a in $\mathcal{A}$ and the $R\exists_{\geq n}$-rule is applicable to a. This contradicts the assumption that $\mathcal{A}$ is complete. (2) a is blocked by an individual c but the same argument as in case (1) holds and leads to the same contradiction.

For $C = \exists_{\leq n} R$ we show the goal by contradiction. Suppose that $a \notin (\exists_{\leq n} R)^{\mathcal{I}_\mathcal{A}}$. Then there exist at least $n+1$ distinct individuals $b_1, \ldots, b_{n+1}$ such that $(a, b_i) \in R^{\mathcal{I}_\mathcal{A}}$, $i \in 1..n+1$. According to Definition 19 the following two cases can occur. (1) We have $n+1$ $(a, b_i) : S_i \in \mathcal{A}$ with $S_i \in R^{\downarrow}$ and $S_i \notin T$, $i \in 1..n+1$. The $R\exists_{\leq n}$ rule cannot be applicable since $\mathcal{A}$ is complete and the b_i are distinct, i.e. $b_i \neq b_j \in \mathcal{A}$, $i, j \in 1..n+1$, $i \neq j$. This contradicts the assumption that $\mathcal{A}$ is clash-free. (2) There exists a blocking individual c with $(c, b_i) : S_i \in \mathcal{A}$, $S_i \in R^{\downarrow}$, and $S_i \notin T$, $i \in 1..n+1$. This leads to an analogous contradiction.

If $\forall x . x : D \in \mathcal{A}$, then –due to the completeness of $\mathcal{A}$– for each individual a in $\mathcal{A}$ we have $a : D \in \mathcal{A}$ and, by the previous cases, $a \in D^{\mathcal{I}_\mathcal{A}}$. Thus, $\mathcal{I}_\mathcal{A}$ satisfies $\forall x . x : D$. Finally, since $\mathcal{I}_\mathcal{A}$ satisfies all assertions in $\mathcal{A}$, $\mathcal{I}_\mathcal{A}$ satisfies $\mathcal{A}$. □

Theorem 21 (Completeness) Let $\mathcal{A}$ be a satisfiable ABox, then there exists at least one completion of $\mathcal{A}$ computed by applying the completion rules.

Proof. Obviously, an Abox containing a clash is unsatisfiable. If every completion of $\mathcal{A}$ is unsatisfiable, then it follows from Proposition 12 that ABox $\mathcal{A}$ is unsatisfiable. □

Definition 22 For any augmentation of an initial ABox $\mathcal{A}$, we define the *concept size* $n_\mathcal{A}$ as the number of concepts or subconcepts occurring in $\mathcal{A}$.[4] Note that $n_\mathcal{A}$ is bound by the length of the string expressing $\mathcal{A}$. The *size* of an ABox $\mathcal{A}$ is defined as $n_\mathcal{A} \times \|T\| + \|O_O\|$.

Lemma 23 Let $\mathcal{A}$ be an ABox and let $\mathcal{A}'$ be a completion of $\mathcal{A}$. In any set X consisting of individuals occurring in $\mathcal{A}'$ with a cardinality greater than $2^{n_\mathcal{A}}$ there exist at least two individuals $a, b \in X$ whose concept sets are equal ($a \equiv_{\mathcal{A}'} b$).

Proof. Each assertion $a : C_i \in \mathcal{A}'$ may contain at most $n_\mathcal{A}$ different concepts C_i. So there cannot exist more than $2^{n_\mathcal{A}}$ different concept sets for the individuals in $\mathcal{A}'$. □

Lemma 24 Let $\mathcal{A}$ be an ABox and let $\mathcal{A}'$ be a completion of $\mathcal{A}$. Then there occur at most $2^{n_\mathcal{A}}$ non-blocked new individuals in $\mathcal{A}'$.

Proof. Suppose we have $2^{n_\mathcal{A}} + 1$ non-blocked new individuals in $\mathcal{A}'$. From Lemma 23 we know that there exist at least two individuals a, b in $\mathcal{A}'$ such that $a \equiv_{\mathcal{A}'} b$. By Definition 13 we have either $a \prec b$ or $b \prec a$. Assume without loss of generality that $a \prec b$ holds and $a \equiv_{\mathcal{A}'} b$ implies $\sigma(\mathcal{A}', a) \supseteq \sigma(\mathcal{A}', b)$. Then we have either $a \succcurlyeq_{\mathcal{A}'} b$ or there exists an individual c with $c \succcurlyeq_{\mathcal{A}'} b$ and $c \prec a$. Both cases contradict the hypothesis. □

Theorem 25 (Termination) Let $\mathcal{A}_\mathcal{T}$ be the augmented ABox w.r.t a TBox $\mathcal{T}$ and let n be the size of $\mathcal{A}_\mathcal{T}$. Every completion of $\mathcal{A}_\mathcal{T}$ is finite and its size is $O(2^{4n})$.

Proof. Let $\mathcal{A}'$ be a completion of $\mathcal{A}_\mathcal{T}$. From Lemma 24 we know that $\mathcal{A}'$ has at most 2^n non-blocked new individuals. Therefore, a total of at most $m \times 2^n$ new individuals may exists in $\mathcal{A}'$, where m is the maximum number of direct successors for any individual in $\mathcal{A}'$.

Note that m is bound by the number of $\exists R . C$ concepts ($\leq n$) plus the total sum of numbers occurring in $\exists_{\geq n} R$. Since numbers are expressed in binary, their sum is bound by 2^n. Hence, we have $m \leq 2^n + n$. Since the number of individuals in the initial ABox is also bound by n, the total number of individuals in $\mathcal{A}'$ is at most $m \times (2^n + n) \leq (2^n + n) \times (2^n + n)$, i.e. $O(2^{2n})$.

The number of different assertions of the form $a : C$ or $\forall x . x : C$ in which each individual in $\mathcal{A}'$ can be involved, is bound by n and each assertion has a size linear in n. Hence, the total size of these assertions is bound $n \times n \times 2^{2n}$, i.e. $O(2^{3n})$.

The number of different assertions of the form $(a, b) : R$ or $a \neq b$ is bound by $(2^{2n})^2$, i.e. $O(2^{4n})$. In conclusion, we have a size of $O(2^{4n})$ for $\mathcal{A}'$. □

Theorem 26 (Decidability) Let $\mathcal{A}_\mathcal{T}$ be an ABox w.r.t. a TBox $\mathcal{T}$. Checking whether $\mathcal{A}_\mathcal{T}$ is satisfiable is a decidable problem.

Proof. This follows immediately from the Theorems 20, 21, and 25. □

5 Practical Reasoning with RACE

The tableaux calculus introduced in the previous sections is of theoretical interest for proving the decidability of the ABox consistency problem. For practical purposes such calculi are highly inefficient. Therefore, the development of optimization techniques is a very important research topic. In order to support practical ABox reasoning with $\mathcal{ALCNH}_{R^+}$ and to empirically

[4] We have to increase $n_\mathcal{A}$ by 1 if $\top$ does not occur in $\mathcal{A}$.

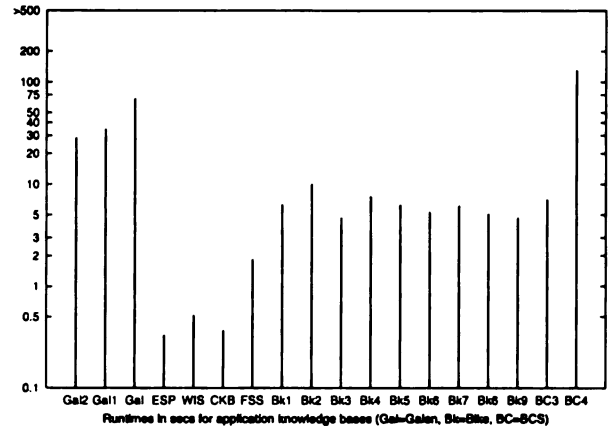


Figure 2: Runtimes for application KBs.

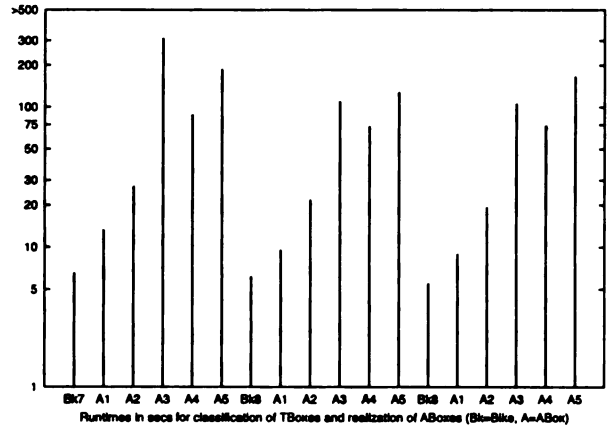


Figure 3: Runtimes for Bike TBoxes and ABoxes.

evaluate optimization techniques for this tableaux calculus, the DL system RACE[5] has been developed [6]. RACE implements an $\mathcal{ALCNH}_{R+}$ reasoner for answering queries concerning ABoxes and TBoxes. It is a successor of HAM-ALC [3]. The RACE architecture incorporates established and novel optimization techniques for TBox and ABox reasoning [6, 5].

The combined effectiveness of these and other techniques are demonstrated with knowledge bases (KBs) derived from actual applications (Figure 2) and a set of ABox benchmark problems (Figures 3, 4). The 'Galen' application KBs are described in [8]. Their employed DLs range from $\mathcal{ALE}$ to $\mathcal{ALCHf}_{R+}$. The KBs 'ESPR', 'WISBER', 'CKB', and 'FSS' (using $\mathcal{ALCNH}$ with GCIs) are taken from previous DL benchmarks but role hierarchies and domain and/or range restrictions for primitive roles (using GCIs) are restored [4]. The

[5]RACE is available from the URL http://kogs-www.informatik.uni-hamburg.de/~moeller/race.html

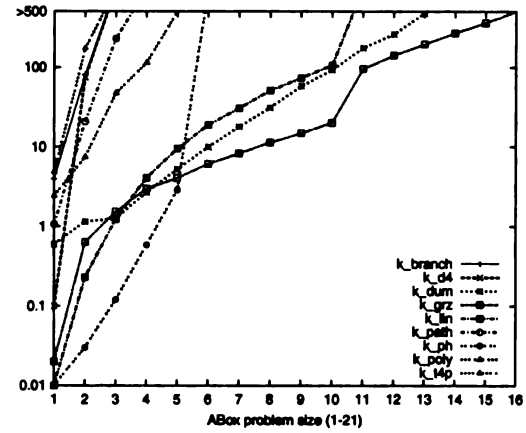


Figure 4: Runtimes for synthetic ABoxes.

'Bike' KBs (using $\mathcal{ALCNH}$ with GCIs) contain configuration knowledge about various types of bicycles. Their corresponding ABoxes describe example configurations of bikes. The KBs 'BCS3' and 'BCS4' (using $\mathcal{ALC}$ with GCIs) are derived from a telecommunication application. Their characteristics is the heavy use of terminological cycles and GCIs.

A set of five ABoxes is iteratively realized using the KBs 'bike7-9' (see Figure 3). The ABoxes describe bike example configurations and exemplify typical classification tasks. The TBoxes 'bike7-9' are almost identical except that they vary in the degree of specifying disjointness between atomic concepts. Figure 4 reports on the runtimes for realizing synthetic ABoxes with an increasing level of difficulty (1-21, see [6] for further explanations).

6 Conclusion

We presented the first treatment for a tableaux calculus deciding the ABox consistency problem for the description logic $\mathcal{ALCNH}_{R+}$. A highly optimized variant of this calculus is already implemented in the ABox description logic system RACE demonstrating the practical usefulness of $\mathcal{ALCNH}_{R+}$. Although TBox reasoners for logics such as $\mathcal{ALCQHI}_{R+}$ are available, the development of $\mathcal{ALCNH}_{R+}$ and its optimized implementation in RACE is a novel approach. Practical reasoning is only possible with the design and implementation of appropriate optimization techniques. This is supported by recent empirical findings suggesting that RACE dramatically outperforms other known DL reasoners for logics at least as expressive as $\mathcal{ALCNH}_{R+}$. To the best of our knowledge there currently exists no other ABox DL system with a performance comparable to RACE.

Acknowledgements

We are grateful to Carsten Lutz, Stephan Tobies, and Michael Wessel for detailed and thoughtful comments on an earlier version of this paper. Recently, in a parallel development, a calculus for deciding the ABox consistency problem for the description logic $\mathcal{ALCQHI}_{R+}$ has been proposed in a revised version of [10].

References

[1] M. Buchheit, F.M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

[2] G. De Giacomo and M. Lenzerini. TBox and ABox reasoning in expressive description logics. In L.C. Aiello, J. Doyle, and S. Shapiro, editors, *Fifth International Conference on Principles of Knowledge Representation, Cambridge, Mass., Nov. 5-8, 1996*, November 1996.

[3] V. Haarslev and R. Möller. Applying an $\mathcal{ALC}$ ABox consistency tester to modal logic SAT problems. In Neil V. Murray, editor, *Proceedings, International Conference on Automatic Reasoning with Analytic Tableaux and Related Methods, TABLEAUX'99, Saratoga Springs, NY, USA*, number 1617 in Lecture Notes in Artificial Intelligence, pages 24–28. Springer Verlag, Berlin, June 1999.

[4] V. Haarslev and R. Möller. An empirical evaluation of optimization strategies for ABox reasoning in expressive description logics. In Lambrix et al. [12], pages 115–119.

[5] V. Haarslev and R. Möller. Optimization techniques for reasoning with expressive ABox and concept expressions. Technical Report FBI-HH-M-290/00, University of Hamburg, Computer Science Department, 2000. Available at URL http://kogs-www.informatik.uni-hamburg.de/~haarslev/publications/report-FBI-290-2000.ps.gz.

[6] V. Haarslev, R. Möller, and A.-Y. Turhan. RACE user's guide and reference manual version 1.1. Technical Report FBI-HH-M-289/99, University of Hamburg, Computer Science Department, October 1999. Available at URL http://kogs-www.informatik.uni-hamburg.de/~haarslev/publications/report-FBI-289-99.ps.gz.

[7] B. Hollunder. *Algorithmic Foundations of Terminological Knowledge Representation Systems*. PhD thesis, University of Saarbrücken, Department of Computer Science, 1994.

[8] I. Horrocks. Using an expressive description logic: FaCT or fiction? In T. Cohn, L. Schubert, and S. Shapiro, editors, *Proceedings of Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Trento, Italy, June 2-5, 1998*, pages 636–647, June 1998.

[9] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3):385–410, June 1999.

[10] I. Horrocks, U. Sattler, and S. Tobies. A description logic with transitive and converse roles, role hierarchies and qualifying number restrictions. Technical Report LTCS-99-08, RWTH Aachen, 1999.

[11] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In Harald Ganzinger, David McAllester, and Andrei Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, September 1999.

[12] P. Lambrix et al., editor. *Proceedings of the International Workshop on Description Logics (DL'99), July 30 - August 1, 1999, Linköping, Sweden*, June 1999.

[13] H.R. Lewis and C.H. Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[14] U. Sattler. A concept language extended with different kinds of transitive roles. In G. Görz and S. Hölldobler, editors, *20. Deutsche Jahrestagung für Künstliche Intelligenz*, number 1137 in Lecture Notes in Artificial Intelligence, pages 333–345. Springer Verlag, Berlin, 1996.

[15] M. Schmidt-Schauss and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[16] S. Tessaris and G. Gough. ABox reasoning with transitive roles and axioms. In Lambrix et al. [12], pages 101–104.

Reasoning with Axioms: Theory and Practice

Ian Horrocks
Department of Computer Science
University of Manchester, UK
horrocks@cs.man.ac.uk

Stephan Tobies
LuFg Theoretical Computer Science
RWTH Aachen, Germany
tobies@informatik.rwth-aachen.de

Abstract

When reasoning in description, modal or temporal logics it is often useful to consider axioms representing universal truths in the domain of discourse. Reasoning with respect to an arbitrary set of axioms is hard, even for relatively inexpressive logics, and it is essential to deal with such axioms in an efficient manner if implemented systems are to be effective in real applications. This is particularly relevant to Description Logics, where subsumption reasoning with respect to a terminology is a fundamental problem. Two optimisation techniques that have proved to be particularly effective in dealing with terminologies are lazy unfolding and absorption. In this paper we seek to improve our theoretical understanding of these important techniques. We define a formal framework that allows the techniques to be precisely described, establish conditions under which they can be safely applied, and prove that, provided these conditions are respected, subsumption testing algorithms will still function correctly. These results are used to show that the procedures used in the FaCT system are correct and, moreover, to show how efficiency can be significantly improved, while still retaining the guarantee of correctness, by relaxing the safety conditions for absorption.

1 MOTIVATION

Description Logics (DLs) form a family of formalisms which have grown out of knowledge representation techniques using frames and semantic networks. DLs use a class based paradigm, describing the domain of interest in terms of concepts (classes) and roles (binary relations) which can be combined using a range of operators to form more complex structured concepts [BHH+91]. A DL *terminology* typically consists of a set of asserted facts, in particular asserted subsumption (is-a-kind-of) relationships between (possibly complex) concepts.[1].

One of the distinguishing characteristics of DLs is a formally defined semantics which allows the structured objects they describe to be reasoned with. Of particular interest is the computation of implied subsumption relationships between concepts, based on the assertions in the terminology, and the maintenance of a concept hierarchy (partial ordering) based on the subsumption relationship [WS92].

The problem of computing concept subsumption relationships has been the subject of much research, and sound and complete algorithms are now known for a wide range of DLs (for example [HN90, BH91, Baa91, DM98, HST99]). However, in spite of the fundamental importance of terminologies in DLs, most of these algorithms deal only with the problem of deciding subsumption between two concepts (or, equivalently, concept satisfiability), without reference to a terminology (but see [BDS93, Cal96, DDM96, HST99]). By restricting the kinds of assertion that can appear in a terminology, concepts can be syntactically expanded so as to explicitly include all relevant terminological information. This procedure, called *unfolding*, has mostly been applied to less expressive DLs. With more expressive DLs, in particular those supporting universal roles, it is often possible to encapsulate an arbitrary terminology in a single concept. This technique can be used with satisfiability testing to ensure that the result is valid with respect to the assertions in the ter-

[1] DLs can also deal with assertions about individuals, but in this paper we will only be concerned with *terminological* (concept based) reasoning

minology, a procedure called *internalisation*.

Although the above mentioned techniques suffice to demonstrate the theoretical adequacy of satisfiability decision procedures for terminological reasoning, experiments with implementations have shown that, for reasons of (lack of) efficiency, they are highly unsatisfactory as a practical methodology for reasoning with DL terminologies. Firstly, experiments with the KRIS system have shown that integrating unfolding with the (tableaux) satisfiability algorithm (*lazy unfolding*) leads to a significant improvement in performance [BFH+94]. More recently, experiments with the FaCT system have shown that reasoning becomes hopelessly intractable when internalisation is used to deal with larger terminologies [Hor98]. However, the FaCT system has also demonstrated that this problem can be dealt with (at least for realistic terminologies) by using a combination of lazy unfolding and internalisation, having first manipulated the terminology in order to minimise the number of assertions that must be dealt with by internalisation (a technique called *absorption*).

It should be noted that, although these techniques were discovered while developing DL systems, they are applicable to a whole range of reasoning systems, independent of the concrete logic and type of algorithm. As well as tableaux based decision procedures, this includes resolution based algorithms, where the importance of minimising the number of terminological sentences has already been noted [HS99], and sequent calculus algorithms, where there is a direct correspondence with tableaux algorithms [BFH+99].

In this paper we seek to improve our theoretical understanding of these important techniques which has, until now, been very limited. In particular we would like to know exactly when and how they can be applied, and be sure that the answers we get from the algorithm are still correct. This is achieved by defining a formal framework that allows the techniques to be precisely described, establishing conditions under which they can be safely applied, and proving that, provided these conditions are respected, satisfiability algorithms will still function correctly. These results are then used to show that the procedures used in the FaCT system are correct[2] and, moreover, to show how efficiency can be significantly improved, while still retaining the guarantee of correctness, by relaxing the safety conditions for absorption. Finally, we identify several interesting directions for future research, in particular the problem of finding the "best" absorption possible.

[2]Previously, the correctness of these procedures had only been demonstrated by a relatively ad-hoc argument [Hor97].

2 PRELIMINARIES

Firstly, we will establish some basic definitions that clarify what we mean by a DL, a terminology (subsequently called a TBox), and subsumption and satisfiability with respect to a terminology, . The results in this paper are uniformly applicable to a whole range of DLs, as long as some basic criteria are met:

Definition 2.1 (Description Logic) *Let* L *be a DL based on infinite sets of atomic concepts* NC *and atomic roles* NR. *We will identify* L *with the sets of its well-formed concepts and require* L *to be closed under boolean operations and sub-concepts.*

An interpretation is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, *where* $\Delta^{\mathcal{I}}$ *is a non-empty set, called the* domain *of* $\mathcal{I}$, *and* $\cdot^{\mathcal{I}}$ *is a function mapping* NC *to* $2^{\Delta^{\mathcal{I}}}$ *and* NR *to* $2^{\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}}$. *With each DL* L *we associate a set* $\mathsf{Int(L)}$ *of* admissible *interpretations for* L. $\mathsf{Int(L)}$ *must be closed under isomorphisms, and, for any two interpretations* $\mathcal{I}$ *and* $\mathcal{I}'$ *that agree on* NR, *it must satisfy* $\mathcal{I} \in \mathsf{Int(L)} \Leftrightarrow \mathcal{I}' \in \mathsf{Int(L)}$. *Additionally, we assume that each DL* L *comes with a semantics that allows any interpretation* $\mathcal{I} \in \mathsf{Int(L)}$ *to be extended to each concept* $C \in \mathsf{L}$ *such that it satisfies the following conditions:*

(I1) it maps the boolean combination of concepts to the corresponding boolean combination of their interpretations, and

(I2) the interpretation $C^{\mathcal{I}}$ *of a compound concept* $C \in \mathsf{L}$ *depends only on the interpretation of those atomic concepts and roles that appear syntactically in* C.

This definition captures a whole range of DLs, namely, the important DL $\mathcal{ALC}$ [SS91] and its many extensions. $\mathsf{Int(L)}$ hides restrictions on the interpretation of certain roles like transitivity, functionality, or role hierarchies, which are imposed by more expressive DLs (e.g., [HST99]), as these are irrelevant for our purposes. In these cases, $\mathsf{Int(L)}$ will only contain those interpretations which interpret the roles as required by the semantics of the logic, e.g., features by partial functions or transitively closed roles by transitive relations. Please note that various modal logics [Sch91], propositional dynamic logics [DL94] and temporal logics [EH85] also fit into this framework. We will use $C \rightarrow D$ as an abbreviation for $\neg C \sqcup D$, $C \leftrightarrow D$ as an abbreviation for $(C \rightarrow D) \sqcap (D \rightarrow C)$, and $\top$ as

a tautological concept, e.g., $A \sqcup \neg A$ for an arbitrary $A \in \mathsf{NC}$.

A TBox consists of a set of axioms asserting subsumption or equality relations between (possibly complex) concepts.

Definition 2.2 (TBox, Satisfiability) *A* TBox $\mathcal{T}$ *for* L *is a finite set of axioms of the form* $C_1 \sqsubseteq C_2$ *or* $C_1 \doteq C_2$, *where* $C_i \in \mathsf{L}$. *If, for some* $A \in \mathsf{NC}$, $\mathcal{T}$ *contains one or more axioms of the form* $A \sqsubseteq C$ *or* $A \doteq C$, *then we say that* A *is* defined *in* $\mathcal{T}$.

Let L *be a DL and* $\mathcal{T}$ *a TBox. An interpretation* $\mathcal{I} \in \mathsf{Int}(\mathsf{L})$ *is a* model *of* $\mathcal{T}$ *iff, for each* $C_1 \sqsubseteq C_2 \in \mathcal{T}$, $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ *holds, and, for each* $C_1 \doteq C_2 \in \mathcal{T}$, $C_1^{\mathcal{I}} = C_2^{\mathcal{I}}$ *holds. In this case we write* $\mathcal{I} \models \mathcal{T}$. *A concept* $C \in \mathsf{L}$ *is* satisfiable *with respect to a TBox* $\mathcal{T}$ *iff there is an* $\mathcal{I} \in \mathsf{Int}(\mathsf{L})$ *with* $\mathcal{I} \models \mathcal{T}$ *and* $C^{\mathcal{I}} \neq \emptyset$. *A concept* $C \in \mathsf{L}$ subsumes *a concept* $D \in \mathsf{L}$ *w.r.t.* $\mathcal{T}$ *iff, for all* $\mathcal{I} \in \mathsf{Int}(\mathsf{L})$ *with* $\mathcal{I} \models \mathcal{T}$, $C^{\mathcal{I}} \supseteq D^{\mathcal{I}}$ *holds.*

Two TBoxes $\mathcal{T}, \mathcal{T}'$ *are called* equivalent *(*$\mathcal{T} \equiv \mathcal{T}'$*), iff, for all* $\mathcal{I} \in \mathsf{Int}(\mathsf{L})$, $\mathcal{I} \models \mathcal{T}$ iff $\mathcal{I} \models \mathcal{T}'$.

We will only deal with concept satisfiability as concept subsumption can be reduced to it for DLs that are closed under boolean operations: C subsumes D w.r.t. $\mathcal{T}$ iff $(D \sqcap \neg C)$ is not satisfiable w.r.t. $\mathcal{T}$.

For temporal or modal logics, satisfiability with respect to a set of formulae $\{C_1, \ldots, C_k\}$ asserted to be universally true corresponds to satisfiability w.r.t. the TBox $\{\top \doteq C_1, \ldots, \top \doteq C_n\}$.

Many decision procedures for DLs base their judgement on the existence of models or pseudo-models for concepts. A central rôle in these algorithms is played by a structure that we will call a *witness* in this paper. It generalises the notions of *tableaux* that appear in DL tableau-algorithms [HNS90, BBH96, HST99] as well as the *Hintikka-structures* that are used in tableau and automata-based decision procedures for temporal logic [EH85] and propositional dynamic logic [VW86].

Definition 2.3 (Witness) *Let* L *be a DL and* $C \in \mathsf{L}$ *a concept. A* witness $\mathcal{W} = (\Delta^{\mathcal{W}}, \cdot^{\mathcal{W}}, \mathcal{L}^{\mathcal{W}})$ *for* C *consists of a non-empty set* $\Delta^{\mathcal{W}}$, *a function* $\cdot^{\mathcal{W}}$ *that maps* NR *to* $2^{\Delta^{\mathcal{W}} \times \Delta^{\mathcal{W}}}$, *and a function* $\mathcal{L}^{\mathcal{W}}$ *that maps* $\Delta^{\mathcal{W}}$ *to* 2^{L} *such that the following properties are satisfied:*

(W1) there is some $x \in \Delta^{\mathcal{W}}$ *with* $C \in \mathcal{L}^{\mathcal{W}}(x)$,

(W2) there is an interpretation $\mathcal{I} \in \mathsf{Int}(\mathsf{L})$ *that* stems *from* $\mathcal{W}$, *and*

(W3) for each interpretation $\mathcal{I} \in \mathsf{Int}(\mathsf{L})$ *that* stems *from* $\mathcal{W}$, *it holds that* $D \in \mathcal{L}^{\mathcal{W}}(x)$ *implies* $x \in D^{\mathcal{I}}$.

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *is said to stem from* $\mathcal{W}$ *if it satisfies:*

1. $\Delta^{\mathcal{I}} = \Delta^{\mathcal{W}}$,
2. $\cdot^{\mathcal{I}}|_{\mathsf{NR}} = \cdot^{\mathcal{W}}$, *and*
3. *for each* $A \in \mathsf{NC}$, $A \in \mathcal{L}^{\mathcal{W}}(x) \Rightarrow x \in A^{\mathcal{I}}$ *and* $\neg A \in \mathcal{L}^{\mathcal{W}}(x) \Rightarrow x \notin A^{\mathcal{I}}$.

A witness $\mathcal{W}$ *is called* admissible *with respect to a TBox* $\mathcal{T}$ *if there is an interpretation* $\mathcal{I} \in \mathsf{Int}(\mathsf{L})$ *that stems from* $\mathcal{W}$ *with* $\mathcal{I} \models \mathcal{T}$.

Please note that, for any witness $\mathcal{W}$, (W2) together with Condition 3 of "stemming" implies that, there exists no $x \in \Delta^{\mathcal{W}}$ and $A \in \mathsf{NC}$, such that $\{A, \neg A\} \subseteq \mathcal{L}^{\mathcal{W}}(x)$. Also note that, in general, more than one interpretation may stem from a witness. This is the case if, for an atomic concept $A \in \mathsf{NC}$ and an element $x \in \Delta^{\mathcal{W}}$, $\mathcal{L}^{\mathcal{W}}(x) \cap \{A, \neg A\} = \emptyset$ holds (because two interpretations $\mathcal{I}$ and $\mathcal{I}'$, with $x \in A^{\mathcal{I}}$ and $x \in \neg A^{\mathcal{I}'}$, could both stem from $\mathcal{W}$).

Obviously, each interpretation $\mathcal{I}$ gives rise to a special witness, called the *canonical witness*:

Definition 2.4 (Canonical Witness) *Let* L *be a DL. For any interpretation* $\mathcal{I} \in \mathsf{Int}(\mathsf{L})$ *we define the* canonical witness $\mathcal{W}_{\mathcal{I}} = (\Delta^{\mathcal{W}_{\mathcal{I}}}, \cdot^{\mathcal{W}_{\mathcal{I}}}, \mathcal{L}^{\mathcal{W}_{\mathcal{I}}})$ *as follows:*

$$\Delta^{\mathcal{W}_{\mathcal{I}}} = \Delta^{\mathcal{I}}$$
$$\cdot^{\mathcal{W}_{\mathcal{I}}} = \cdot^{\mathcal{I}}|_{\mathsf{NR}}$$
$$\mathcal{L}^{\mathcal{W}_{\mathcal{I}}} = \lambda x.\{D \in \mathsf{L} \mid x \in D^{\mathcal{I}}\}$$

The following elementary properties of a canonical witness will be useful in our considerations.

Lemma 2.5 *Let* L *be a DL,* $C \in \mathsf{L}$, *and* $\mathcal{T}$ *a TBox. For each* $\mathcal{I} \in \mathsf{Int}(\mathsf{L})$ *with* $C^{\mathcal{I}} \neq \emptyset$,

1. *each interpretation* $\mathcal{I}'$ *stemming from* $\mathcal{W}_{\mathcal{I}}$ *is isomorphic to* $\mathcal{I}$
2. $\mathcal{W}_{\mathcal{I}}$ *is a witness for* C,
3. $\mathcal{W}_{\mathcal{I}}$ *is admissible w.r.t.* $\mathcal{T}$ *iff* $\mathcal{I} \models \mathcal{T}$

Proof.

1. Let $\mathcal{I}'$ stem from $\mathcal{W}_{\mathcal{I}}$. This implies $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}$ and $\cdot^{\mathcal{I}'}|_{\mathsf{NR}} = \cdot^{\mathcal{I}}|_{\mathsf{NR}}$. For each $x \in \Delta^{\mathcal{I}}$ and $A \in \mathsf{NC}$, $\{A, \neg A\} \cap \mathcal{L}^{\mathcal{W}_{\mathcal{I}}}(x) \neq \emptyset$, this implies $\cdot^{\mathcal{I}'}|_{\mathsf{NC}} = \cdot^{\mathcal{I}}|_{\mathsf{NC}}$ and hence $\mathcal{I}$ and $\mathcal{I}'$ are isomorphic.

2. Properties (W1) and (W2) hold by construction. Obviously, $\mathcal{I}$ stems from $\mathcal{W}_\mathcal{I}$ and from (1) it follows that each interpretation $\mathcal{I}'$ stemming from $\mathcal{W}_\mathcal{I}$ is isomorphic to $\mathcal{I}$, hence (W3) holds.

3. Since $\mathcal{I}$ stems from $\mathcal{W}_\mathcal{I}$, $\mathcal{I} \models \mathcal{T}$ implies that $\mathcal{W}_\mathcal{I}$ is admissible w.r.t. $\mathcal{T}$. If $\mathcal{W}_\mathcal{I}$ is admissible w.r.t. $\mathcal{T}$, then there is an interpretation $\mathcal{I}'$ stemming from $\mathcal{W}_\mathcal{I}$ with $\mathcal{I}' \models \mathcal{T}$. Since $\mathcal{I}$ is isomorphic to $\mathcal{I}'$, this implies $\mathcal{I} \models \mathcal{T}$. ∎

As a corollary we get that the existence of admissible witnesses is closely related to the satisfiability of concepts w.r.t. TBoxes:

Lemma 2.6 *Let* L *be a DL. A concept* $C \in \mathsf{L}$ *is satisfiable w.r.t. a TBox* $\mathcal{T}$ *iff it has a witness that is admissible w.r.t.* $\mathcal{T}$.

Proof. For the *only if*-direction let $\mathcal{I} \in \mathsf{Int}(\mathsf{L})$ be an interpretation with $\mathcal{I} \models \mathcal{T}$ and $C^\mathcal{I} \neq \emptyset$. From Lemma 2.5 it follows that the canonical witness $\mathcal{W}_\mathcal{I}$ is a witness for C that is admissible w.r.t. $\mathcal{T}$.

For the *if*-direction let $\mathcal{W}$ be an witness for C that is admissible w.r.t. $\mathcal{T}$. This implies that there is an interpretation $\mathcal{I} \in \mathsf{Int}(\mathsf{L})$ stemming from $\mathcal{W}$ with $\mathcal{I} \models \mathcal{T}$. For each interpretation $\mathcal{I}$ that stems from $\mathcal{W}$, it holds that $C^\mathcal{I} \neq \emptyset$ due to (W1) and (W3). ∎

From this it follows that one can test the satisfiability of a concept w.r.t. to a TBox by checking for the existence of an admissible witness. We call algorithms that utilise this approach *model-building algorithms*.

This notion captures tableau-based decision procedures, [HNS90, BBH96, HST99], those using automata-theoretic approaches [VW86, CDL99] and, due to their direct correspondence with tableaux algorithms [HS99, BFH+99], even resolution based and sequent calculus algorithms.

The way many decision procedures for DLs deal with TBoxes exploits the following simple lemma.

Lemma 2.7 *Let* L *be a DL,* $C \in \mathsf{L}$ *a concept, and* $\mathcal{T}$ *a TBox. Let* $\mathcal{W}$ *be a witness for* C. *If*

$$\begin{array}{lcl} C_1 \sqsubseteq C_2 \in \mathcal{T} & \Rightarrow & \forall x \in \Delta^\mathcal{W}.(C_1 \to C_2 \in \mathcal{L}^\mathcal{W}(x)) \\ C_1 \doteq C_2 \in \mathcal{T} & \Rightarrow & \forall x \in \Delta^\mathcal{W}.(C_1 \leftrightarrow C_2 \in \mathcal{L}^\mathcal{W}(x)) \end{array}$$

then $\mathcal{W}$ *is admissible w.r.t.* $\mathcal{T}$.

Proof. $\mathcal{W}$ is a witness, hence there is an interpretation $\mathcal{I} \in \mathsf{Int}(\mathsf{L})$ stemming from $\mathcal{W}$. From (W3) and the fact that $\mathcal{W}$ satisfies the properties stated in 2.7 it follows that, for each $x \in \Delta^\mathcal{I}$,

$$\begin{array}{lcl} C_1 \sqsubseteq C_2 \in \mathcal{T} & \Rightarrow & C_1 \to C_2 \in \mathcal{L}^\mathcal{W}(x) \\ & \Rightarrow & x \in (C_1 \to C_2)^\mathcal{I} \\ C_1 \doteq C_2 \in \mathcal{T} & \Rightarrow & C_1 \leftrightarrow C_2 \in \mathcal{L}^\mathcal{W}(x) \\ & \Rightarrow & x \in (C_1 \leftrightarrow C_2)^\mathcal{I} \end{array}$$

Hence, $\mathcal{I} \models \mathcal{T}$ and $\mathcal{W}$ is admissible w.r.t. $\mathcal{T}$. ∎

Examples of algorithms that exploit this lemma to deal with axioms can be found in [DDM96, DL96, HST99], where, for each axiom $C_1 \sqsubseteq C_2$ ($C_1 \doteq C_2$) the concept $C_1 \to C_2$ ($C_1 \leftrightarrow C_2$) is added to every node of the generated tableau.

Dealing with general axioms in this manner is costly due to the high degree of nondeterminism introduced. This can best be understood by looking at tableaux algorithms, which try to build witnesses in an incremental fashion. For a concept C to be tested for satisfiability, they start with $\Delta^\mathcal{W} = \{x_0\}$, $\mathcal{L}^\mathcal{W}(x_0) = \{C\}$ and $\cdot^\mathcal{W}(R) = \emptyset$ for each $R \in \mathsf{NR}$. Subsequently, the concepts in $\mathcal{L}^\mathcal{W}$ are decomposed and, if necessary, new nodes are added to $\Delta^\mathcal{W}$, until either $\mathcal{W}$ is a witness for C, or an obvious contradiction of the form $\{A, \neg A\} \subseteq \mathcal{L}^\mathcal{W}(x)$, which violates (W2), is generated. In the latter case, backtracking search is used to explore alternative non-deterministic decompositions (e.g., of disjunctions), one of which could lead to the discovery of a witness.

When applying Lemma 2.7, disjunctions are added to the label of each node of the tableau for each general axiom in the TBox (one disjunction for axioms of the form $C_1 \sqsubseteq C_2$, two for axioms of the form $C_1 \doteq C_2$). This leads to an exponential increase in the search space as the number of nodes and axioms increases. For example, with 10 nodes and a TBox containing 10 general axioms (of the form $C_1 \sqsubseteq C_2$) there are already 100 disjunctions, and they can be nondeterministically decomposed in 2^{100} different ways. For a TBox containing large numbers of general axioms (there are 1,214 in the GALEN medical terminology KB [RNG93]), this can degrade performance to the extent that subsumption testing is effectively non-terminating. To reason with this kind of TBox we must find a more efficient way to deal with axioms.

3 ABSORPTIONS

We start our considerations with an analysis of a technique that can be used to deal more efficiently with so-called primitive or acyclic TBoxes.

Definition 3.1 (Absorption) *Let* L *be a DL and* $\mathcal{T}$ *a TBox. An* absorption *of* $\mathcal{T}$ *is a pair of TBoxes*

$(\mathcal{T}_u, \mathcal{T}_g)$ such that $\mathcal{T} \equiv \mathcal{T}_u \cup \mathcal{T}_g$ and $\mathcal{T}_u$ contains only axioms of the form $A \sqsubseteq D$ and $\neg A \sqsubseteq D$ where $A \in \mathsf{NC}$.

An absorption $(\mathcal{T}_u, \mathcal{T}_g)$ of $\mathcal{T}$ is called correct *if it satisfies the following condition. For each witness $\mathcal{W}$, if, for each $x \in \Delta^{\mathcal{W}}$,*

$$A \sqsubseteq D \in \mathcal{T}_u \wedge A \in \mathcal{L}^{\mathcal{W}}(x) \Rightarrow D \in \mathcal{L}^{\mathcal{W}}(x)$$
$$\neg A \sqsubseteq D \in \mathcal{T}_u \wedge \neg A \in \mathcal{L}^{\mathcal{W}}(x) \Rightarrow D \in \mathcal{L}^{\mathcal{W}}(x)$$
$$C_1 \sqsubseteq C_2 \in \mathcal{T}_g \Rightarrow C_1 \rightarrow C_2 \in \mathcal{L}^{\mathcal{W}}(x)$$
$$C_1 \doteq C_2 \in \mathcal{T}_g \Rightarrow C_1 \leftrightarrow C_2 \in \mathcal{L}^{\mathcal{W}}(x)$$

then $\mathcal{W}$ is admissible w.r.t. $\mathcal{T}$. We refer to this properties by $(*)$. A witness that satisfies $(*)$ will be called unfolded w.r.t. $\mathcal{T}$.

If the reference to a specific TBox is clear from the context, we will often leave the TBox implicit and say that a witness is unfolded.

How does a correct absorption enable an algorithm to deal with axioms more efficiently? This is best described by returning to tableaux algorithms. Instead of dealing with axioms as previously described, which may lead to an exponential increase in the search space, axioms in $\mathcal{T}_u$ can now be dealt with in a deterministic manner. Assume, for example, that we have to handle the axiom $A \doteq C$. If the label of a node already contains A (resp. $\neg A$), then C (resp. $\neg C$) is added to the label; if the label contains neither A nor $\neg A$, then *nothing* has to be done. Dealing with the axioms in $\mathcal{T}_u$ this way avoids the necessity for additional non-deterministic choices and leads to a gain in efficiency. A witness produced in this manner will be unfolded and is a certificate for satisfiability w.r.t. $\mathcal{T}$. This technique is generally known as *lazy unfolding* of primitive TBoxes [Hor98]; formally, it is justified by the following lemma:

Lemma 3.2 *Let $(\mathcal{T}_u, \mathcal{T}_g)$ be a correct absorption of $\mathcal{T}$. For any $C \in \mathsf{L}$, C has a witness that is admissible w.r.t. $\mathcal{T}$ iff C has an unfolded witness.*

Proof. The *if*-direction follows from the definition of "correct absorption". For the *only if*-direction, let $C \in \mathsf{L}$ be a concept and $\mathcal{W}$ a witness for C that is admissible w.r.t. $\mathcal{T}$. This implies the existence of an interpretation $\mathcal{I} \in \mathsf{Int}(\mathsf{L})$ stemming from $\mathcal{W}$ such that $\mathcal{I} \models \mathcal{T}$ and $C^{\mathcal{I}} \neq \emptyset$. Since $\mathcal{T} \equiv \mathcal{T}_u \cup \mathcal{T}_g$ we have $\mathcal{I} \models \mathcal{T}_u \cup \mathcal{T}_g$ and hence the canonical witness $\mathcal{W}_{\mathcal{I}}$ is an unfolded witness for C. ∎

A family of TBoxes where absorption can successfully be applied are *primitive* TBoxes, the most simple form of TBox usually studied in the literature.

Definition 3.3 (Primitive TBox) *A TBox $\mathcal{T}$ is called* primitive *iff it consists entirely of axioms of the form $A \doteq D$ with $A \in \mathsf{NC}$, each $A \in \mathsf{NC}$ appears as at most one left-hand side of an axiom, and $\mathcal{T}$ is acyclic. Acyclicity is defined as follows: $A \in \mathsf{NC}$ is said to* directly use *$B \in \mathsf{NC}$ if $A \doteq D \in \mathcal{T}$ and B occurs in D;* uses *is the transitive closure of "directly uses". We say that $\mathcal{T}$ is* acyclic *if there is no $A \in \mathsf{NC}$ that uses itself.*

For primitive TBoxes a correct absorption can easily be given.

Theorem 3.4 *Let $\mathcal{T}$ be a primitive TBox, $\mathcal{T}_g = \emptyset$, and $\mathcal{T}_u$ defined by*

$$\mathcal{T}_u = \{A \sqsubseteq D, \neg A \sqsubseteq \neg D \mid A \doteq D \in \mathcal{T}\}.$$

Then $(\mathcal{T}_u, \mathcal{T}_g)$ is a correct absorption of $\mathcal{T}$.

Proof. Trivially, $\mathcal{T} \equiv \mathcal{T}_u \cup \mathcal{T}_g$ holds. Given an unfolded witness $\mathcal{W}$, we have to show that there is an interpretation $\mathcal{I}$ stemming from $\mathcal{W}$ with $\mathcal{I} \models \mathcal{T}$.

We fix an arbitrary linearisation $A_1, \ldots, A_k$ of the "uses" partial order on the atomic concept names appearing on the left-hand sides of axioms in $\mathcal{T}$ such that, if A_i uses A_j, then $j < i$ and the defining concept for A_i is D_i.

For some interpretation $\mathcal{I}$, atomic concept A, and set $X \subseteq \Delta^{\mathcal{I}}$, we denote the interpretation that maps A to X and agrees with $\mathcal{I}$ on all other atomic concepts and roles by $\mathcal{I}[A \mapsto X]$. For $0 \leq i \leq k$, we define $\mathcal{I}_i$ in an iterative process starting from an arbitrary interpretation $\mathcal{I}_0$ stemming from $\mathcal{W}$ and setting

$$\mathcal{I}_i := \mathcal{I}_{i-1}[A_i \mapsto \{x \in \Delta^{\mathcal{W}} \mid x \in D_i^{\mathcal{I}_{i-1}}\}]$$

Since, for each A_i there is exactly one axiom in $\mathcal{T}$, each step in this process is well-defined. Also, since $\mathsf{Int}(\mathsf{L})$ may only restrict the interpretation of atomic roles, $\mathcal{I}_i \in \mathsf{Int}(\mathsf{L})$ for each $0 \leq i \leq k$. For $\mathcal{I} = \mathcal{I}_k$ it can be shown that $\mathcal{I}$ is an interpretation stemming from $\mathcal{W}$ with $\mathcal{I} \models \mathcal{T}$.

First we prove inductively that, for $0 \leq i \leq k$, $\mathcal{I}_i$ stems from $\mathcal{W}$. We have already required $\mathcal{I}_0$ to stem from $\mathcal{W}$.

Assume the claim was proved for $\mathcal{I}_{i-1}$ and $\mathcal{I}_i$ does not stem from $\mathcal{W}$. Then there must be some $x \in \Delta^{\mathcal{W}}$ such that either (i) $A_i \in \mathcal{L}^{\mathcal{W}}(x)$ but $x \notin A_i^{\mathcal{I}_i}$ or (ii) $\neg A_i \in \mathcal{L}^{\mathcal{W}}(x)$ but $x \in A_i^{\mathcal{I}_i}$ (since we assume $\mathcal{I}_{i-1}$ to stem from $\mathcal{W}$ and A_i is the only atomic concept whose interpretation changes from $\mathcal{I}_{i-1}$ to $\mathcal{I}_i$). The two cases can be handled dually:

(i) From $A_i \in \mathcal{L}^{\mathcal{W}}(x)$ it follows that $D_i \in \mathcal{L}^{\mathcal{W}}(x)$, because $\mathcal{W}$ is unfolded. Since $\mathcal{I}_{i-1}$ stems from $\mathcal{W}$ and $\mathcal{W}$ is a witness, Property (W3) implies $x \in D_i^{\mathcal{I}_{i-1}}$. But this implies $x \in A_i^{\mathcal{I}_i}$, which is a contradiction.

(ii) From $\neg A_i \in \mathcal{L}^{\mathcal{W}}(x)$ it follows that $\neg D_i \in \mathcal{L}^{\mathcal{W}}(x)$ because $\mathcal{W}$ is unfolded. Since $\mathcal{I}_{i-1}$ stems from $\mathcal{W}$ and $\mathcal{W}$ is an witness, Property (W3) implies $x \in (\neg D_i)^{\mathcal{I}_{i-1}}$. Since $(\neg D_i)^{\mathcal{I}_{i-1}} = \Delta^{\mathcal{W}} \setminus D_i^{\mathcal{I}_{i-1}}$ this implies $x \notin A_i^{\mathcal{I}_i}$, which is a contradiction.

Together this implies that $\mathcal{I}_i$ also stems from $\mathcal{W}$.

To show that $\mathcal{I} \models \mathcal{T}$ we show inductively that $\mathcal{I}_i \models A_j \doteq D_j$ for each $1 \leq j \leq i$. This is obviously true for $i = 0$.

The interpretation of D_i may not depend on the interpretation of A_i because otherwise (I2) would imply that A_i uses itself. Hence $D_i^{\mathcal{I}_i} = D_i^{\mathcal{I}_{i-1}}$ and, by construction, $\mathcal{I}_i \models A_i \doteq D_i$. Assume there is some $j < i$ such that $\mathcal{I}_i \not\models A_j \doteq D_j$. Since $\mathcal{I}_{i-1} \models A_j \doteq D_j$ and only the interpretation of A_i has changed from $\mathcal{I}_{i-1}$ to $\mathcal{I}_i$, $D_j^{\mathcal{I}_i} \neq D_j^{\mathcal{I}_{i-1}}$ must hold because of (I2). But this implies that A_i occurs in D_j and hence A_j uses A_i which contradicts $j < i$. Thus, we have $\mathcal{I} \models A_j = D_j$ for each $1 \leq j \leq k$ and hence $\mathcal{I} \models \mathcal{T}$. ∎

Lazy unfolding is a well-known and widely used technique for optimising reasoning w.r.t. primitive TBoxes [BFH+94]. So far, we have only given a correctness proof for this relatively simple approach, although one that is independent of a specific DL or reasoning algorithm. With the next lemma we show how we can extend correct absorptions and hence how lazy unfolding can be applied to a broader class of TBoxes. A further enhancement of the technique is presented in Section 5.

Lemma 3.5 *Let $(\mathcal{T}_u, \mathcal{T}_g)$ be a correct absorption of a TBox $\mathcal{T}$.*

1. *If $\mathcal{T}'$ is an arbitrary TBox, then $(\mathcal{T}_u, \mathcal{T}_g \cup \mathcal{T}')$ is a correct absorption of $\mathcal{T} \cup \mathcal{T}'$.*
2. *If $\mathcal{T}'$ is a TBox that consists entirely of axioms of the form $A \sqsubseteq D$, where $A \in \mathsf{NC}$ and A is not defined in $\mathcal{T}_u$, then $(\mathcal{T}_u \cup \mathcal{T}', \mathcal{T}_g)$ is a correct absorption of $\mathcal{T} \cup \mathcal{T}'$.*

Proof. In both cases, $\mathcal{T}_u \cup \mathcal{T}_g \cup \mathcal{T}' \equiv \mathcal{T} \cup \mathcal{T}'$ holds trivially.

1. Let $C \in \mathsf{L}$ be a concept and $\mathcal{W}$ be an unfolded witness for C w.r.t. the absorption $(\mathcal{T}_u, \mathcal{T}_g \cup \mathcal{T}')$. This implies that $\mathcal{W}$ is unfolded w.r.t. the (smaller) absorption $(\mathcal{T}_u, \mathcal{T}_g)$. Since $(\mathcal{T}_u, \mathcal{T}_g)$ is a correct absorption, there is an interpretation $\mathcal{I}$ stemming from $\mathcal{W}$ with $\mathcal{I} \models \mathcal{T}$. Assume $\mathcal{I} \not\models \mathcal{T}'$. Then, without loss of generality,[3] there is an axiom $D \sqsubseteq E \in \mathcal{T}'$ such that there exists an $x \in D^{\mathcal{I}} \setminus E^{\mathcal{I}}$. Since $\mathcal{W}$ is unfolded, we have $D \to E \in \mathcal{L}^{\mathcal{W}}(x)$ and hence (W3) implies $x \in (\neg D \sqcup E)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus (D^{\mathcal{I}} \setminus E^{\mathcal{I}})$, a contradiction. Hence $\mathcal{I} \models \mathcal{T} \cup \mathcal{T}'$ and $\mathcal{W}$ is admissible w.r.t. $\mathcal{T} \cup \mathcal{T}'$.

2. Let $C \in \mathsf{L}$ be a concept and $\mathcal{W}$ be an unfolded witness for C w.r.t. the absorption $(\mathcal{T}_u \cup \mathcal{T}', \mathcal{T}_g)$. From $\mathcal{W}$ we define a new witness $\mathcal{W}'$ for C by setting $\Delta^{\mathcal{W}'} := \Delta^{\mathcal{W}}$, $\cdot^{\mathcal{W}'} := \cdot^{\mathcal{W}}$, and definig $\mathcal{L}^{\mathcal{W}'}$ to be the function that, for every $x \in \Delta^{\mathcal{W}'}$, maps x to the set

$$\mathcal{L}^{\mathcal{W}}(x) \cup \{\neg A \mid A \sqsubseteq D \in \mathcal{T}', A \notin \mathcal{L}^{\mathcal{W}}(x)\}$$

It is easy to see that $\mathcal{W}'$ is indeed a witness for C and that $\mathcal{W}'$ is also unfolded w.r.t. the absorption $(\mathcal{T}_u \cup \mathcal{T}', \mathcal{T}_g)$. This implies that $\mathcal{W}'$ is also unfolded w.r.t. the (smaller) absorption $(\mathcal{T}_u, \mathcal{T}_g)$. Since $(\mathcal{T}_u, \mathcal{T}_g)$ is a correct absorption of $\mathcal{T}$, there exists an interpretation $\mathcal{I}$ stemming from $\mathcal{W}'$ such that $\mathcal{I} \models \mathcal{T}$. We will show that $\mathcal{I} \models \mathcal{T}'$ also holds. Assume $\mathcal{I} \not\models \mathcal{T}'$, then there is an axiom $A \sqsubseteq D \in \mathcal{T}'$ and an $x \in \Delta^{\mathcal{I}}$ such that $x \in A^{\mathcal{I}}$ but $x \notin D^{\mathcal{I}}$. By construction of $\mathcal{W}'$, $x \in A^{\mathcal{I}}$ implies $A \in \mathcal{L}^{\mathcal{W}'}(x)$ because otherwise $\neg A \in \mathcal{L}^{\mathcal{W}'}(x)$ would hold in contradiction to (W3). Then, since $\mathcal{W}'$ is unfolded, $D \in \mathcal{L}^{\mathcal{W}'}(x)$, which, again by (W3), implies $x \in D^{\mathcal{I}}$, a contradiction.

Hence, we have shown that there exists an interpretation $\mathcal{I}$ stemming from $\mathcal{W}'$ such that $\mathcal{I} \models \mathcal{T}_u \cup \mathcal{T}' \cup \mathcal{T}_g$. By construction of $\mathcal{W}'$, any interpretation stemming from $\mathcal{W}'$ also stems from $\mathcal{W}$, hence $\mathcal{W}$ is admissible w.r.t. $\mathcal{T} \cup \mathcal{T}'$. ∎

4 APPLICATION TO FaCT

In the preceeding section we have defined correct absorptions and discussed how they can be exploited in order to optimise satisfiability procedures. However, we have said nothing about the problem of how to find an absorption given an arbitrary terminology. In this section we will describe the absorption algorithm used by FaCT and prove that it generates correct absorptions.

[3] Arbitrary TBoxes can be expressed using only axioms of the form $C \sqsubseteq D$.

Given a TBox $\mathcal{T}$ containing arbitrary axioms, the absorption algorithm used by FaCT constructs a triple of TBoxes $(\mathcal{T}_g, \mathcal{T}_{\text{prim}}, \mathcal{T}_{\text{inc}})$ such that

- $\mathcal{T} \equiv \mathcal{T}_g \cup \mathcal{T}_{\text{prim}} \cup \mathcal{T}_{\text{inc}}$,
- $\mathcal{T}_{\text{prim}}$ is primitive, and
- $\mathcal{T}_{\text{inc}}$ consists only of axioms of the form $A \sqsubseteq D$ where $A \in \mathsf{NC}$ and A is not defined in $\mathcal{T}_{\text{prim}}$.

We refer to these properties by $(*)$. From Theorem 3.4 together with Lemma 3.5 it follows that, for

$$\mathcal{T}_u := \{A \sqsubseteq D, \neg A \sqsubseteq \neg D \mid A \doteq D \in \mathcal{T}_{\text{prim}}\} \cup \mathcal{T}_{\text{inc}}$$

$(\mathcal{T}_u, \mathcal{T}_g)$ is a correct absorption of $\mathcal{T}$; hence satisfiability for a concept C w.r.t. $\mathcal{T}$ can be decided by checking for an unfolded witness for C.

In a first step, FaCT distributes axioms from $\mathcal{T}$ amongst $\mathcal{T}_{\text{inc}}$, $\mathcal{T}_{\text{prim}}$, and $\mathcal{T}_g$, trying to minimise the number of axioms in $\mathcal{T}_g$ while still maintaining $(*)$. To do this, it initialises $\mathcal{T}_{\text{prim}}, \mathcal{T}_{\text{inc}}$, and $\mathcal{T}_g$ with $\emptyset$, and then processes each axiom $X \in \mathcal{T}$ as follows.

1. If X is of the form $A \sqsubseteq C$, then
   (a) if $A \in \mathsf{NC}$ and A is not defined in $\mathcal{T}_{\text{prim}}$ then X is added to $\mathcal{T}_{\text{inc}}$,
   (b) otherwise X is added to $\mathcal{T}_g$
2. If X is of the form $A \doteq C$, then
   (a) if $A \in \mathsf{NC}$, A is not defined in $\mathcal{T}_{\text{prim}}$ or $\mathcal{T}_{\text{inc}}$ and $\mathcal{T}_{\text{prim}} \cup \{X\}$ is primitive, then X is added to $\mathcal{T}_{\text{prim}}$,
   (b) otherwise, the axioms $A \sqsubseteq C$ and $C \sqsubseteq A$ are added to $\mathcal{T}_g$

It is easy to see that the resulting TBoxes $\mathcal{T}_g, \mathcal{T}_{\text{prim}}, \mathcal{T}_{\text{inc}}$ satisfy $(*)$. In a second step, FaCT processes the axioms in $\mathcal{T}_g$ one at a time, trying to absorb them into axioms in $\mathcal{T}_{\text{inc}}$. Those axioms that are not absorbed remain in $\mathcal{T}_g$. To give a simpler formulation of the algorithm, each axiom $(C \sqsubseteq D) \in \mathcal{T}_g$ is viewed as a clause $\mathbf{G} = \{D, \neg C\}$, corresponding to the axiom $\top \sqsubseteq C \to D$, which is equivalent to $C \sqsubseteq D$. For each such axiom FaCT applies the following absorption procedure.

1. Try to absorb $\mathbf{G}$. If there is a concept $\neg A \in \mathbf{G}$ such that $A \in \mathsf{NC}$ and A is not defined in $\mathcal{T}_{\text{prim}}$, then add $A \sqsubseteq B$ to $\mathcal{T}_{\text{inc}}$, where B is the disjunction of all the concepts in $\mathbf{G} \setminus \{\neg A\}$, remove $\mathbf{G}$ from $\mathcal{T}_g$, and exit.
2. Try to simplify $\mathbf{G}$.
   (a) If there is some $\neg C \in \mathbf{G}$ such that C is of the form $C_1 \sqcap \ldots \sqcap C_n$, then substitute $\neg C$ with $\neg C_1 \sqcup \ldots \sqcup \neg C_n$, and continue with step 2b.
   (b) If there is some $C \in \mathbf{G}$ such that C is of the form $(C_1 \sqcup \ldots \sqcup C_n)$, then apply associativity by setting $\mathbf{G} = \mathbf{G} \cup \{C_1, \ldots, C_n\} \setminus \{(C_1 \sqcup \ldots \sqcup C_n)\}$, and return to step 1.
3. Try to unfold $\mathbf{G}$. If, for some $A \in \mathbf{G}$ (resp. $\neg A \in \mathbf{G}$), there is an axiom $A \doteq C$ in $\mathcal{T}_{\text{prim}}$, then substitute $A \in \mathbf{G}$ (resp. $\neg A \in \mathbf{G}$) with C (resp. $\neg C$) and return to step 1.
4. If none of the above were possible, then absorption of $\mathbf{G}$ has failed. Leave $\mathbf{G}$ in $\mathcal{T}_g$, and exit.

For each step, we have to show that $(*)$ is maintained. Dealing with clauses instead of axioms causes no problems. In the first step, axioms are moved from $\mathcal{T}_g$ to $\mathcal{T}_{\text{inc}}$ as long as this does not violate $(*)$. The second and the third step replace a clause by an equivalent one and hence do not violate $(*)$.

Termination of the procedure is obvious. Each axiom is considered only once and, for a given axiom, simplification and unfolding can only be applied finitely often before the procedure is exited, either by absorbing the axiom into $\mathcal{T}_{\text{inc}}$ or leaving it in $\mathcal{T}_g$. For simplification, this is obvious; for unfolding, this holds because $\mathcal{T}_{\text{prim}}$ is acyclic. Hence, we get the following:

Theorem 4.1 *For any TBox $\mathcal{T}$, FaCT computes a correct absorption of $\mathcal{T}$.*

5 IMPROVING PERFORMANCE

The absorption algorithm employed by FaCT already leads to a dramatic improvement in performance. This is illustrated by Figure 1, which shows the times taken by FaCT to classify versions of the GALEN KB with some or all of the general axioms removed. Without absorption, classification time increased rapidly with the number of general axioms, and exceeded 10,000s with only 25 general axioms in the KB; with absorption, only 160s was taken to classify the KB with all 1,214 general axioms.

However, there is still considerable scope for further gains. In particular, the following definition for a *stratified* TBox allows lazy unfolding to be more generally applied, while still allowing for correct absorptions.

Definition 5.1 (Stratified TBox) *A TBox $\mathcal{T}$ is called* stratified *iff it consists entirely of axioms of the*

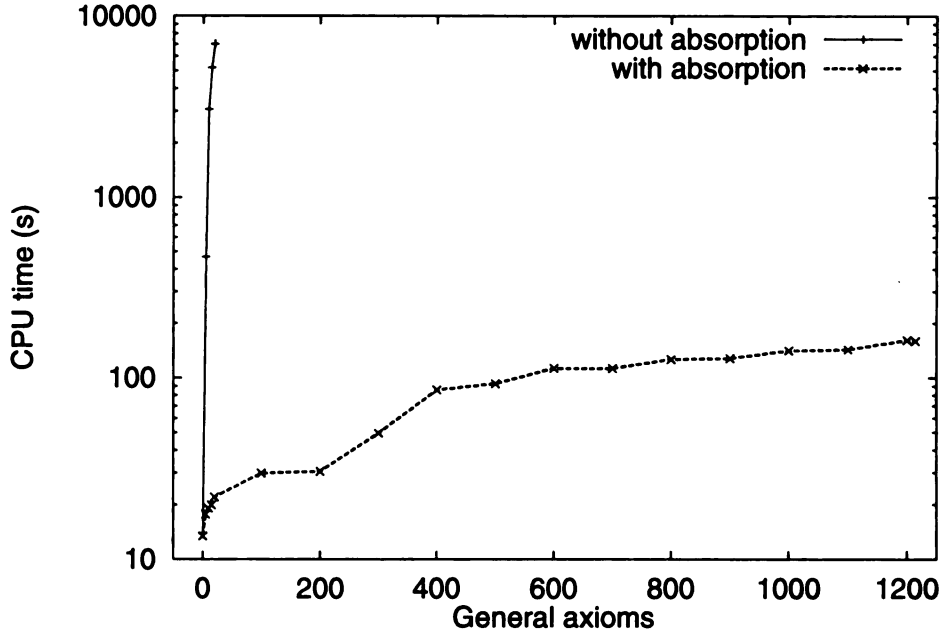


Figure 1: Classification times with and without absorption

form $A \doteq D$ with $A \in \mathsf{NC}$, each $A \in \mathsf{NC}$ appears at most once on the left-hand side of an axiom, and $\mathcal{T}$ can be arranged monotonously, i.e., there is a disjoint partition $\mathcal{T}_1 \dot\cup \mathcal{T}_2 \dot\cup \ldots \dot\cup \mathcal{T}_k$ of $\mathcal{T}$, such that

- *for all $1 \leq j < i \leq k$, if $A \in \mathsf{NC}$ is defined in $\mathcal{T}_i$, then it does not occur in $\mathcal{T}_j$, and*
- *for all $1 \leq i \leq k$, all concepts which appear on the right-hand side of axioms in $\mathcal{T}_i$ are monotone in all atomic concepts defined in $\mathcal{T}_i$.*

A concept C is monotone in an atomic concept A if, for any interpretation $\mathcal{I} \in \mathsf{Int(L)}$ and any two sets $X_1, X_2 \subseteq \Delta^{\mathcal{I}}$,

$$X_1 \subseteq X_2 \;\Rightarrow\; C^{\mathcal{I}[A \mapsto X_1]} \subseteq C^{\mathcal{I}[A \mapsto X_2]}.$$

For many DLs, a sufficient condition for monotonicity is *syntactic* monotonicity, i.e., a concept C is syntactically monotone in some atomic concept A if A does no appear in C in the scope of an odd number of negations.

Obviously, due to its acyclicity, every primitive TBox is also stratified and hence the following theorem is a strict generalisation of Theorem 3.4.

Theorem 5.2 *Let $\mathcal{T}$ be a stratified TBox, $\mathcal{T}_g = \emptyset$ and $\mathcal{T}_u$ defined by*

$$\mathcal{T}_u = \{A \sqsubseteq D, \neg A \sqsubseteq \neg D \mid A \doteq D \in \mathcal{T}\}.$$

Then $(\mathcal{T}_u, \mathcal{T}_g)$ is a correct absorption of $\mathcal{T}$.

The proof of this theorem follows the same line as the proof of Theorem 3.4. Starting from an arbitrary interpretation $\mathcal{I}_0$ stemming from the unfolded witness, we incrementally construct interpretations $\mathcal{I}_1, \ldots, \mathcal{I}_k$, using a fixed point construction in each step. We show that each $\mathcal{I}_i$ stems from $\mathcal{W}$ and that, for $1 \leq j < i \leq k$, $\mathcal{I}_i \models \mathcal{T}_j$, hence $\mathcal{I}_k \models \mathcal{T}$ and stems from $\mathcal{W}$.

Before we prove this theorem, we recall some basics of lattice theory. For any set $\mathcal{S}$, the powerset of $\mathcal{S}$, denoted by $2^{\mathcal{S}}$ forms a complete lattice, where the ordering, join and meet operations are set-inclusion $\subseteq$, union $\cup$, and intersection $\cap$, respectively. For any complete lattice $\mathcal{L}$, its n-fold cartesian product $\mathcal{L}^n$ is also a complete lattice, with ordering, join, and meet defined in a pointwise manner.

For a lattice $\mathcal{L}$, a function $\Phi : \mathcal{L} \to \mathcal{L}$ is called monotone, iff, for $x_1, x_2 \in \mathcal{L}$, $x_1 \sqsubseteq x_2$ implies $\Phi(x_1) \sqsubseteq \Phi(x_2)$.

By Tarski's fixed point theorem [Tar55], every monotone function Φ on a complete lattice, has uniquely defined least and greatest fixed points, i.e., there are elements $\overline{x}, \underline{x} \in \mathcal{L}$ such that

$$\overline{x} = \Phi(\overline{x}) \text{ and } \underline{x} = \Phi(\underline{x})$$

and, for all $x \in \mathcal{L}$ with $x = \Phi(x)$,

$$\underline{x} \sqsubseteq x \text{ and } x \sqsubseteq \overline{x}.$$

Proof of Theorem 5.2. $\mathcal{T}_u \cup \mathcal{T}_g \equiv \mathcal{T}$ is obvious. Let $\mathcal{W} = (\Delta^{\mathcal{W}}, \cdot^{\mathcal{W}}, \mathcal{L}^{\mathcal{W}})$ be an unfolded witness. We have to show that there is an interpretation $\mathcal{I}$ stemming from $\mathcal{W}$ with $\mathcal{I} \models \mathcal{T}$. Let $\mathcal{T}_1, \ldots, \mathcal{T}_k$ be the required partition of $\mathcal{T}$. We will define $\mathcal{I}$ inductively, starting with an arbitrary interpretation $\mathcal{I}_0$ stemming from $\mathcal{W}$.

Assume $\mathcal{I}_{i-1}$ was already defined. We define $\mathcal{I}_i$ from $\mathcal{I}_{i-1}$ as follows: let $\{A_1^i \doteq D_1^i, \ldots, A_m^i \doteq D_m^i\}$ be an enumeration of $\mathcal{T}_i$. First we need some auxiliary notation: for any concept $C \in \mathsf{L}$ we define

$$C^{\mathcal{W}} := \{x \in \Delta^{\mathcal{W}} \mid C \in \mathcal{L}^{\mathcal{W}}(x)\}.$$

Using this notation we define the function Φ mapping subsets $X_1, \ldots, X_m$ of $\Delta^{\mathcal{W}}$ to

$$\begin{array}{l}(((A_1^i)^{\mathcal{W}} \cup (D_1^i)^{\mathcal{I}_{i-1}(X_1,\ldots,X_m)}) \setminus (\neg A_1^i)^{\mathcal{W}}, \\ \ldots, \\ ((A_m^i)^{\mathcal{W}} \cup (D_m^i)^{\mathcal{I}_{i-1}(X_1,\ldots,X_m)}) \setminus (\neg A_m^i)^{\mathcal{W}})\end{array}$$

where

$$\mathcal{I}_{i-1}(X_1, \ldots, X_m) := \mathcal{I}_{i-1}[A_1^i \mapsto X_1, \ldots, A_m^i \mapsto X_m]$$

Since all of the D_j^i are monotone in all of the A_m^i, Φ is a monontone function. This implies that Φ has a least fixed point, which we denote by $(\underline{X}_1, \ldots, \underline{X}_m)$. We use this fixed point to define $\mathcal{I}_i$ by

$$\mathcal{I}_i := \mathcal{I}_{i-1}[A_1^i \mapsto \underline{X}_1, \ldots, A_m^i \mapsto \underline{X}_m]$$

CLAIM 1: For each $0 \leq i \leq k$, $\mathcal{I}_i$ stems from $\mathcal{W}$.

We show this claim by induction on i. We have already required $\mathcal{I}_0$ to stem from $\mathcal{W}$. Assume $\mathcal{I}_{i-1}$ stems from $\mathcal{W}$. Since the only thing that changes from $\mathcal{I}_{i-1}$ to $\mathcal{I}_i$ is the interpretation of the atomic concepts $A_1^i, \ldots, A_m^i$, we only have to check that $A_j^i \in \mathcal{L}^{\mathcal{W}}(x)$ implies $x \in (A_j^i)^{\mathcal{I}_i}$ and $\neg A_j^i \in \mathcal{L}^{\mathcal{W}}(x)$ implies $x \notin (A_j^i)^{\mathcal{I}_i}$.

By definition of Φ, and because $\{x \mid A_j^i \in \mathcal{L}^{\mathcal{W}}(x)\} \cap \{x \mid \neg A_j^i \in \mathcal{L}^{\mathcal{W}}(x)\} = \emptyset$, $A_j^i \in \mathcal{L}^{\mathcal{W}}(x)$ implies $x \in (A_j^i)^{\mathcal{I}_i}$. Also by the definition of Φ, $\neg A_j^i \in \mathcal{L}^{\mathcal{W}}(x)$ implies $x \notin (A_j^i)^{\mathcal{I}_i}$. Hence, $\mathcal{I}_i$ stems from $\mathcal{W}$.

CLAIM 2: For each $1 \leq j \leq i \leq k$, $\mathcal{I}_i \models \mathcal{T}_j$.

We prove this claim by induction over i starting from 0. For $i = 0$, there is nothing to prove. Assume the claim would hold for $\mathcal{I}_{i-1}$. The only thing that changes from $\mathcal{I}_{i-1}$ to $\mathcal{I}_i$ is the interpretation of the atomic concepts $A_1^i, \ldots A_m^i$ defined in $\mathcal{T}_i$. Since these concepts may not occur in $\mathcal{T}_j$ for $j < i$, the interpretation of the concepts in these TBoxes does not change, and from $\mathcal{I}_{i-1} \models \mathcal{T}_j$ follows $\mathcal{I}_i \models \mathcal{T}_j$ for $1 \leq j \leq i-1$.

It remains to show that $\mathcal{I}_i \models \mathcal{T}_i$. Let $A_j^i \doteq D_j^i$ be an axiom from $\mathcal{T}_i$. From the definition of $\mathcal{I}_i$ we have

$$(A_j^i)^{\mathcal{I}_i} = ((A_j^i)^{\mathcal{W}} \cup (D_j^i)^{\mathcal{I}_i}) \setminus (\neg A_j^i)^{\mathcal{W}}. \quad (1)$$

$\mathcal{W}$ is unfolded, hence $A_j^i \in \mathcal{L}^{\mathcal{W}}(x)$ implies $D_j^i \in \mathcal{L}^{\mathcal{W}}(x)$ and, since $\mathcal{I}_i$ stems from $\mathcal{W}$, this implies $x \in (D_j^i)^{\mathcal{I}_i}$, thus

$$(A_j^i)^{\mathcal{W}} \cup (D_j^i)^{\mathcal{I}_i} = (D_j^i)^{\mathcal{I}_i} \quad (2)$$

Furthermore, $\neg A_j^i \in \mathcal{L}^{\mathcal{W}}(x)$ implies $\neg D_j^i \in \mathcal{L}^{\mathcal{W}}(x)$ implies $x \in (\neg D_j^i)^{\mathcal{I}_i}$, thus

$$(D_j^i)^{\mathcal{I}_i} \setminus (\neg A_j^i)^{\mathcal{W}} = (D_j^i)^{\mathcal{I}_i} \quad (3)$$

Taking together (1), (2), and (3) we get

$$(A_j^i)^{\mathcal{I}_i} = (D_j^i)^{\mathcal{I}_i},$$

and hence $\mathcal{I}_i \models A_j^i \doteq D_j^i$.

Together, Claim 1 and Claim 2 prove the theorem, since $\mathcal{I}_k$ is an interpretation that stems from $\mathcal{W}$ and satisfies $\mathcal{T}$. ∎

This theorem makes it possible to apply the same lazy unfolding strategy as before to cyclical definitions. Such definitions are quite natural in a logic that supports inverse roles. For example, an orthopaedic procedure might be defined as a procedure performed by an orthopaedic surgeon, while an orthopaedic surgeon might be defined as a surgeon who performs only orthopaedic procedures:[4]

$$\begin{array}{c}\text{o-procedure} \doteq \text{procedure} \sqcap (\exists \textit{performs}^{-}.\text{o-surgeon}) \\ \text{o-surgeon} \doteq \text{surgeon} \sqcap (\forall \textit{performs}.\text{o-procedure})\end{array}$$

The absorption algorithm described in Section 4 would force the second of these definitions to be added to $\mathcal{T}_g$ as two general axioms and, although both axioms would subsequently be absorbed into $\mathcal{T}_u$, the procedure would result in a disjunctive term being added to one of the definitions in $\mathcal{T}_u$. Using Theorem 5.2 to enhance the absorption algorithm so that these kinds of definition are directly added to $\mathcal{T}_u$ reduces the number of disjunctive terms in $\mathcal{T}_u$ and can lead to significant improvements in performance.

This can be demonstrated by a simple experiment with the new FaCT system, which implements the $\mathcal{SHIQ}$

[4] This example is only intended for didactic purposes.

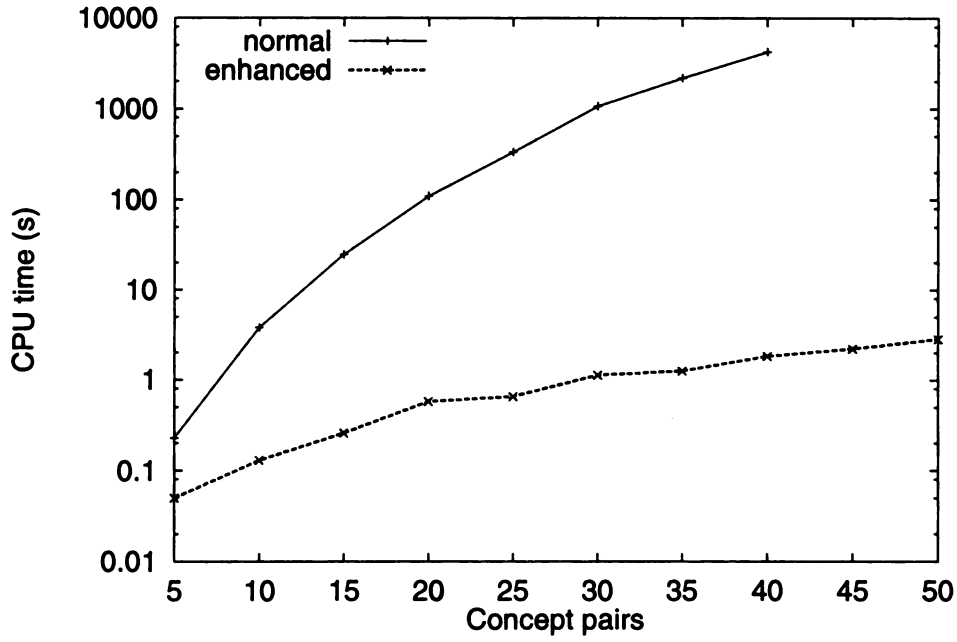


Figure 2: Classification times with and without enhanced absorption

logic [HST99] and is thus able to deal with inverse roles. Figure 2 shows the classification time in seconds using the normal and enhanced absorption algorithms for terminologies consisting of between 5 and 50 pairs of cyclical definitions like those described above for o-surgeon and o-procedure. With only 10 pairs the gain in performance is already a factor of 30, while for 45 and 50 pairs it has reached several orders of magnitude: with the enhanced absorption the terminology is classified in 2–3 seconds whereas with the original algorithm the time required exceeded the 10,000 second limit imposed in the experiment.

It is worth pointing out that it is by no means trivially true that cyclical definitions can be dealt with by lazy unfolding. Even without inverse roles it is clear that definitions such as $A \doteq \neg A$ (or more subtle variants) force the domain to be empty and would lead to an incorrect absorption if dealt with by lazy unfolding. With converse roles it is, for example, possible to force the interpretation of a role R to be empty with a definition such as $A \doteq \forall R.(\forall R^{-}.\neg A)$, again leading to an incorrect absorption if dealt with by lazy unfolding.

6 OPTIMAL ABSORPTIONS

We have demonstrated that absorption is a highly effective and widely applicable technique, and by formally defining correctness criteria for absorptions we have proved that the procedure used by FaCT finds correct absorptions. Moreover, by establishing more precise correctness criteria we have demonstrated how the effectiveness of this procedure could be further enhanced.

However, the absorption algorithm used by FaCT is clearly sub-optimal, in the sense that changes could be made that would, in general, allow more axioms to be absorbed (e.g., by also giving special consideration to axioms of the form $\neg A \sqsubseteq C$ with $A \in \mathsf{NC}$). Moreover, the procedure is non-deterministic, and, while it is guaranteed to produce a correct absorption, its specific result depends on the order of the axioms in the original TBox $\mathcal{T}$. Since the semantics of a TBox $\mathcal{T}$ does not depend on the order of its axioms, there is no reason to suppose that they will be arranged in a way that yields a "good" absorption. Given the effectiveness of absorption, it would be desirable to have an algorithm that was guaranteed to find the "best" absorption possible for any set of axioms, irrespective of their ordering in the TBox.

Unfortunately, it is not even clear how to define a sensible optimality criterion for absorptions. It is obvious that simplistic approaches based on the number or size of axioms remaining in $\mathcal{T}_g$ will not lead to a useful solution for this problem. Consider, for example, the cyclical TBox experiment from the previous section. Both the original FaCT absorption algorithm and the

enhanced algorithm, which exploits Theorem 5.2, are able to compute a complete absorption of the axioms (i.e., a correct absorption with $\mathcal{T}_g = \emptyset$), but the enhanced algorithm leads to much better performance, as shown in Figure 2.

An important issue for future work is, therefore, the identification of a suitable optimality criterion for absorptions, and the development of an algorithm that is able to compute absorptions that are optimal with respect to this criterion.

Acknowledgements

This work was partially supported by the DFG, Project No. GR 1324/3-1.

References

[Baa91] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proc. of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 446–451, 1991.

[BBH96] F. Baader, M. Buchheit, and B. Hollunder. Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.

[BDS93] M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *J. of Artificial Intelligence Research*, 1:109–138, 1993.

[BFH+94] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems, or: Making KRIS get a move on. *Applied Artificial Intelligence*, 4:109–132, 1994.

[BFH+99] A. Borgida, E. Franconi, I. Horrocks, D. McGuinness, and P. F. Patel-Schneider. Explaining $\mathcal{ALC}$ subsumption. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Proc. of the International Workshop on Description Logics (DL'99)*, pages 37–40, 1999.

[BH91] F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithms. In *Processing declarative knowledge: International workshop PDK'91*, number 567 in Lecture Notes in Artificial Intelligence, pages 67–86, Berlin, 1991. Springer-Verlag.

[BHH+91] F. Baader, H.-J. Heinsohn, B. Hollunder, J. Muller, B. Nebel, W. Nutt, and H.-J. Profitlich. Terminological knowledge representation: A proposal for a terminological logic. Technical Memo TM-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH (DFKI), 1991.

[Cal96] D. Calvanese. Reasoning with inclusion axioms in description logics: Algorithms and complexity. In Wolfgang Wahlster, editor, *Proc. of the 12th European Conference on Artificial Intelligence (ECAI'96)*, pages 303–307. John Wiley & Sons Ltd., 1996.

[CDL99] D. Calvanese, G. De Giacomo, and M. Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, 1999.

[DDM96] F. Donini, G. De Giacomo, and F. Massacci. EXPTIME tableaux for $\mathcal{ALC}$. In L. Padgham, E. Franconi, M. Gehrke, D. L. McGuinness, and P. F. Patel-Schneider, editors, *Collected Papers from the International Description Logics Workshop (DL'96)*, number WS-96-05 in AAAI Technical Report, pages 107–110. AAAI Press, Menlo Park, California, 1996.

[DL94] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI'94)*, pages 205–212. AAAI Press/The MIT Press, 1994.

[DL96] G. De Giacomo and M. Lenzerini. TBox and ABox reasoning in expressive description logics. In Luigia C. Aiello, John Doyle, and Stuart C. Shapiro, editors, *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 316–327. Morgan Kaufmann, Los Altos, 1996.

[DM98] G. De Giacomo and F. Massacci. Combining deduction and model checking into tableaux and algorithms for converse-PDL.

Information and Computation: special issue on the Federated Logic Conferences. Academic Press, 1998.

[EH85] E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *J. of Computer and System Sciences*, 30:1–24, 1985.

[HN90] B. Hollunder and W. Nutt. Subsumption algorithms for concept languages. In *Proc. of the 9th European Conference on Artificial Intelligence (ECAI'90)*, pages 348–353. John Wiley & Sons Ltd., 1990.

[HNS90] B. Hollunder, W. Nutt, and M. Schmidt-Schauss. Subsumption algorithms for concept description languages. In *ECAI-90*, Pitman Publishing, London, 1990.

[Hor97] I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics.* PhD thesis, University of Manchester, 1997.

[Hor98] I. Horrocks. Using an expressive description logic: FaCT or fiction? In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proc. of the Sixth International Conference (KR'98)*, pages 636–647. Morgan Kaufmann Publishers, San Francisco, California, June 1998.

[HS99] U. Hustadt and R. A. Schmidt. On the relation of resolution and tableaux proof systems for description logics. In *Proc. of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 110–115, 1999.

[HST99] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, pages 161–180, 1999.

[RNG93] A. L. Rector, W A Nowlan, and A Glowinski. Goals for concept representation in the GALEN project. In *Proc. of the 17th Annual Symposium on Computer Applications in Medical Care (SCAMC'93)*, pages 414–418, Washington DC, USA, 1993.

[Sch91] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 466–471, Sydney, 1991.

[SS91] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Acta Informatica*, 48(1):1–26, 1991.

[Tar55] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. of Mathematics*, 5:285–309, 1955.

[VW86] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *J. of Computer and System Sciences*, 32:183–221, 1986. A preliminary version appeared in *Proc. of the 16th ACM SIGACT Symp. on Theory of Computing (STOC'84).*

[WS92] W. A. Woods and J. G. Schmolze. The KL-ONE family. *Computers and Mathematics with Applications – Special Issue on Artificial Intelligence*, 23(2–5):133–177, 1992.

Rewriting Concepts Using Terminologies

Franz Baader, Ralf Küsters, and Ralf Molitor
Theoretical Computer Science, RWTH Aachen
Ahornstraße 55, 52074 Aachen, Germany
email: {baader,kuesters,molitor}@informatik.rwth-aachen.de

Abstract

The problem of rewriting a concept given a terminology can informally be stated as follows: given a terminology $\mathcal{T}$ (i.e., a set of concept definitions) and a concept description C that does not contain concept names defined in $\mathcal{T}$, can this description be rewritten into a "related better" description E by using (some of) the names defined in $\mathcal{T}$?

In this paper, we first introduce a general framework for the rewriting problem in description logics, and then concentrate on one specific instance of the framework, namely the minimal rewriting problem (where "better" means shorter, and "related" means equivalent). We investigate the complexity of the decision problem induced by the minimal rewriting problem for the languages $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$, and $\mathcal{ALC}$, and then introduce an algorithm for computing (minimal) rewritings for the language $\mathcal{ALE}$. (In the full paper, a similar algorithm is also developed for $\mathcal{ALN}$.) Finally, we sketch other interesting instances of the framework.

1 Motivation

In description logics (DLs), the standard inference problems, like the subsumption and the instance problem, are now well-investigated. More recently, new types of inference problems have been introduced and investigated, like matching (Borgida and McGuinness, 1996; Baader et al., 1999a; Baader and Küsters, 2000) and computing the least common subsumer (Cohen et al., 1992; Cohen and Hirsh, 1994; Baader and Küsters, 1998; Baader et al., 1999b). In contrast to the standard inferences, algorithms that solve these nonstandard problems produce *concept descriptions as output*, which are then returned to the user for inspection. For example, in an application in chemical process engineering (Baader and Sattler, 1996; Sattler, 1998) we try to support the bottom-up construction of knowledge bases by computing most specific concepts (msc) of individuals and least common subsumers (lcs) of concepts: instead of directly defining a new concept, the knowledge engineer introduces several typical examples as individuals, which are then generalized into a concept description by using the msc and the lcs operation (Baader and Küsters, 1998; Baader et al., 1999b). This description is then offered to the knowledge engineer as a possible candidate for a definition of the concept.

In such a framework, it is important that the returned description is as readable and comprehensible as possible. Unfortunately, the descriptions that are produced by the known algorithms for solving the nonstandard inference problems in general do not satisfy this requirement. The reason is that – like most algorithms for the standard inference problems – these algorithms work on unfolded descriptions, i.e., concept descriptions that do not contain names defined in the underlying terminology (TBox). Consequently, the descriptions that they produce also do not use defined names, which makes them large and hard to read and comprehend. One possibility to overcome this problem would be to modify the known algorithms for the nonstandard inference problems such that they can take defined names into account. In order to avoid having to modify all these algorithms separately, we propose not to change the algorithms themselves, but to add rewriting as a post-processing step to them.

Informally, the problem of rewriting a concept given a terminology can be stated as follows: given a TBox $\mathcal{T}$ (i.e., a set of concept definitions) and a concept description C that does not contain concept names defined in $\mathcal{T}$, can this description be rewrit-

ten into an "related better" description E by using (some of) the names defined in $\mathcal{T}$? In this paper, related will mean equivalent, and better will mean shorter (but one can also imagine other optimality criteria). For example, if $\mathcal{T}$ contains the definition Parent $\doteq$ Human $\sqcap \exists$has-child.Human, then the concept description Human $\sqcap \exists$has-child.(Human $\sqcap \exists$has-child.Human) can be rewritten into the two smaller descriptions Human $\sqcap \exists$has-child.Parent and Parent $\sqcap \exists$has-child.Parent, which are both equivalent to the original description.

The formal framework for rewriting that will be introduced in Section 3 encompasses this type of rewriting (called the minimal rewriting problem in the following), but also has other interesting instances (see Section 7). In Section 4 we investigate the complexity of the decision problem induced by the minimal rewriting problem for the DLs $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$, and $\mathcal{ALC}$. This will show that (unless P = NP) minimal rewritings cannot be computed in polynomial time, even for DLs with a polynomial subsumption problem. Section 5 then sketches an algorithm for computing all minimal rewritings for the DL $\mathcal{ALE}$. (A similar algorithm exists for $\mathcal{ALN}$ (Baader et al., 1999c).) In Section 6, we describe a more efficient rewriting algorithm, which computes one (possibly non-minimal) rewriting using a greedy heuristics. Due to space limitations, we cannot give complete proofs of all the results presented in this paper. All details can, however, be found in the full paper (Baader et al., 1999c).

2 Preliminaries

Concept descriptions are inductively defined with the help of a set of *constructors*, starting with a set N_C of *concept names* and a set N_R of *role names*. In this work, we consider concept descriptions built from the constructors shown in Table 1. The concept descriptions in the description logics $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$, and $\mathcal{ALC}$ are built using certain subsets of these constructors, as shown in the last four columns of Table 1. When talking about an arbitrary DL, we will usually employ the letter $\mathcal{L}$ (possibly with subscript).

The semantics of concept descriptions is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The domain $\Delta^{\mathcal{I}}$ of $\mathcal{I}$ is a non-empty set of individuals and the interpretation function $\cdot^{\mathcal{I}}$ maps each concept name $P \in N_C$ to a set $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and each role name $r \in N_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined, as shown in the third column of Table 1.

The terminology of an application domain can be represented in a so-called TBox. A *TBox* $\mathcal{T}$ is a finite set of *concept definitions* of the form $A \doteq C$, where $A \in N_C$ is a concept name and C is a concept description. The interpretation $\mathcal{I}$ is a model of the TBox $\mathcal{T}$ iff it satisfies $A^{\mathcal{I}} = C^{\mathcal{I}}$ for all concept definitions $A \doteq C$ in $\mathcal{T}$.

The concept name A is a *defined name* in the TBox $\mathcal{T}$ iff it occurs on the left-hand side of a concept definition in $\mathcal{T}$; otherwise, A is called *primitive name*. The concept description C in $A \doteq C$ is called the *defining concept of* A. For a given DL $\mathcal{L}$, we talk about $\mathcal{L}$-concept descriptions and $\mathcal{L}$-TBoxes, if all constructors occurring in the concept descriptions and concept definitions belong to $\mathcal{L}$. Throughout the paper, we assume TBoxes to be (1) without multiple definitions, i.e., for each defined name A, there exists a unique concept definition of the form $A \doteq C$ in $\mathcal{T}$; and (2) acyclic, i.e., the defining concept of a defined name must not, directly or indirectly, refer to this name (see (Nebel, 1990a) for exact definitions). The TBox $\mathcal{T}$ is called *unfolded* iff all defining concepts in $\mathcal{T}$ do not contain defined names (Nebel, 1990a). Because of our assumptions on TBoxes, a given TBox can always be transformed into an equivalent unfolded TBox; however, this unfolding process can lead to an exponential blow-up of the TBox (Nebel, 1990b).

One of the most important inference services provided by DL systems is computing the subsumption hierarchy. The concept description D *subsumes* the concept description C ($C \sqsubseteq D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$; D is *equivalent* to C ($C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$. In the presence of a TBox $\mathcal{T}$, we say that D *subsumes* C *modulo* $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$, and C is *equivalent to* D *modulo* $\mathcal{T}$ ($C \equiv_{\mathcal{T}} D$) iff $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$.

3 A general framework for rewriting in DLs

Definition 1 (Rewriting) *Let N_R be a set of role names and N_P a set of primitive names, and let $\mathcal{L}_s$, $\mathcal{L}_d$, and $\mathcal{L}_t$ be three DLs (the source-, destination, and TBox-DL, respectively). A* rewriting problem *is given by*

- *an $\mathcal{L}_t$-TBox $\mathcal{T}$ containing only role names from N_R and primitive names from N_P; the set of defined names occurring in $\mathcal{T}$ is denoted by N_D;*
- *an $\mathcal{L}_s$-concept description C using only the names from N_R and N_P;*
- *a binary relation $\rho \subseteq \mathcal{L}_s \times \mathcal{L}_d$ between $\mathcal{L}_s$- and $\mathcal{L}_d$-concept descriptions.*

Construct name	Syntax	Semantics	$\mathcal{FL}_0$	$\mathcal{ALE}$	$\mathcal{ALC}$	$\mathcal{ALN}$
Top	$\top$	$\Delta^{\mathcal{I}}$		x	x	x
Bottom	$\bot$	$\emptyset$		x	x	x
Primitive negation ($P \in N_C$)	$\neg P$	$\Delta^{\mathcal{I}} \setminus P^{\mathcal{I}}$		x	x	x
Negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$			x	
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	x	x	x	x
Disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$			x	
Existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$		x	x	
Value restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$	x	x	x	x
At least number restriction	$(\geq n\, r)$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\} \geq n\}$				x
At most number restriction	$(\leq n\, r)$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\} \leq n\}$				x

Table 1: Syntax and semantics of concept descriptions.

An $\mathcal{L}_d$-rewriting of C using $\mathcal{T}$ is an $\mathcal{L}_d$-concept description E built using names from N_R and $N_P \cup N_D$ such that $C \rho E$.

Given an appropriate ordering $\preceq$ on $\mathcal{L}_d$-concepts, a rewriting E is called $\preceq$-minimal *iff there does not exist a rewriting E' such that $E' \prec E$.*

As an example, consider the instance of the framework where all three DLs are the language $\mathcal{ALN}$, the relation ρ is instantiated by equivalence modulo $\mathcal{T}$, and the ordering $\preceq$ is induced by the size of the concept descriptions. Let

$$
\begin{aligned}
C \;=\;& \mathsf{Male} \sqcap \mathsf{Rich} \sqcap (\geq 1\ \mathsf{has\text{-}child}) \sqcap \\
& \forall \mathsf{has\text{-}child}.(\mathsf{Male} \sqcap \mathsf{Rich}), \quad \text{and} \\
\mathcal{T} \;=\;& \{\mathsf{Father} \doteq \mathsf{Male} \sqcap (\geq 1\ \mathsf{has\text{-}child}), \\
& \mathsf{RichParent} \doteq \mathsf{Rich} \sqcap \forall \mathsf{has\text{-}child}.\mathsf{Rich} \sqcap \\
& \qquad (\geq 1\ \mathsf{has\text{-}child}), \\
& \mathsf{FatherOfSons} \doteq \mathsf{Father} \sqcap \forall \mathsf{has\text{-}child}.\mathsf{Male}\}.
\end{aligned}
$$

It is easy to see that the concept description $\mathsf{FatherOfSons} \sqcap \mathsf{RichParent}$ is an $\mathcal{ALN}$-rewriting of C using $\mathcal{T}$, and that its size is minimal.

This was an example of what we will call the *minimal rewriting problem*, i.e., the instance of the framework where (i) all three DLs are the same language $\mathcal{L}$; (ii) the binary relation ρ corresponds to equivalence modulo the TBox; and (iii) $\mathcal{L}$-concept descriptions are ordered by size, i.e., $E \preceq E'$ iff $|E| \leq |E'|$. The size $|E|$ of a concept description E is defined to be the number of occurrences of concept and role names in E (where $\top$ and $\bot$ are *not* counted).

In the present paper, we will restrict our attention to the minimal rewriting problem for the DLs $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$, and $\mathcal{ALC}$. Other interesting instances of the framework will be mentioned in Section 7.

4 The minimal rewriting decision problem

In order to determine the complexity of the minimal rewriting problem, we first consider the *decision problem* induced by this optimization problem for a given DL $\mathcal{L}$: Given an $\mathcal{L}$-concept description C, an $\mathcal{L}$-TBox $\mathcal{T}$, and a nonnegative integer κ, does there exist an $\mathcal{L}$-rewriting E of C using $\mathcal{T}$ such that $|E| \leq \kappa$?

Since this decision problem can obviously be reduced to the problem of computing a minimal rewriting of C using $\mathcal{T}$, hardness results for the decision problem carry over to the optimization problem. In the sequel, we give lower and upper bounds for the complexity of the minimal rewriting decision problem for the DLs $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$, and $\mathcal{ALC}$.

NP-Hardness for $\mathcal{FL}_0$, $\mathcal{ALN}$, and $\mathcal{ALE}$

We give a reduction of the NP-complete problem SETCOVER (Garey and Johnson, 1979) to the minimal rewriting decision problem in $\mathcal{FL}_0$. An instance of the SETCOVER problem is of the following form:

Instance: A finite set $\mathcal{U} = \{u_1, \ldots, u_n\}$, a family $\mathcal{F} = \{F_i \subseteq \mathcal{U} \mid 1 \leq i \leq m\}$ of non-empty subsets of $\mathcal{U}$, and a nonnegative integer κ.

Question: Does there exist a subset $\{F_{i_1}, \ldots, F_{i_k}\}$ of $\mathcal{F}$ of size $k \leq \kappa$ such that $F_{i_1} \cup \ldots \cup F_{i_k} = \mathcal{U}$?

Obviously, we can restrict our attention to instances of the problem where at least $\mathcal{F}$ itself covers $\mathcal{U}$, i.e., $F_1 \cup \ldots \cup F_m = \mathcal{U}$.

For a given instance $(\mathcal{U}, \mathcal{F}, \kappa)$ of the SETCOVER problem, we view $\mathcal{U}$ as set of primitive names, and define the corresponding instance of the minimal rewriting decision problem in $\mathcal{FL}_0$ as follows:

$$C_{\mathcal{U}} \;:=\; u_1 \sqcap \ldots \sqcap u_n$$

$$\mathcal{T}_{\mathcal{F}} := \{A_j \doteq \bigsqcap_{u \in F_j} u \mid 1 \leq j \leq m\}.$$

NP-hardness for the minimal rewriting decision problem in $\mathcal{FL}_0$ is an immediate consequence of the following lemma.

Lemma 2 *There exists a minimal rewriting E of $C_{\mathcal{U}}$ using $\mathcal{T}_{\mathcal{F}}$ with $|E| \leq \kappa$ iff there exists a cover of $\mathcal{U}$ with $k \leq \kappa$ sets from $\mathcal{F}$.*

Proof: A rewriting of $C_{\mathcal{U}}$ of size $k \leq \kappa$ is of the form $D = A_{i_1} \sqcap \ldots \sqcap A_{i_l} \sqcap v_{l+1} \sqcap \ldots \sqcap v_k$ for some $1 \leq l \leq k$ and $v_j \in \mathcal{U}$ (for $l+1 \leq j \leq k$).

First, we show that we can (w.l.o.g.) assume that $l = k$, i.e., D does not contain primitive names. Since $\mathcal{F}$ covers $\mathcal{U}$, we know that for each v_j, $l+1 \leq j \leq k$, there exists $F_{i_j} \in \mathcal{F}$ with $v_j \in F_{i_j}$. Thus, replacing each v_j by A_{i_j} yields a rewriting D' of $C_{\mathcal{U}}$ such that D' does not contain primitive names, and $|D'| \leq |D|$.

Now, let $D = A_{i_1} \sqcap \ldots \sqcap A_{i_k}$ be a rewriting of $C_{\mathcal{U}}$ that does not contain primitive names. Then $C \equiv_{\mathcal{T}} D$ implies that, for each $u \in \mathcal{U}$ there exists a defined name A_{i_j} such that u occurs in the right-hand side of the definition of A_{i_j}. Hence, $F_{i_1} \cup \ldots \cup F_{i_k}$ is a cover of $\mathcal{U}$ of size $k \leq \kappa$.

Conversely, let $F_{i_1} \cup \ldots \cup F_{i_k}$ be a cover of $\mathcal{U}$ of size $k \leq \kappa$. Then $D := A_{i_1} \sqcap \ldots \sqcap A_{i_k}$ is a rewriting of $C_{\mathcal{U}}$ of size $k \leq \kappa$. □

It is easy to see that the above reduction is still valid if we view the concept $C_{\mathcal{U}}$ as $\mathcal{ALN}$- or $\mathcal{ALE}$-concept and the TBox $\mathcal{T}_{\mathcal{F}}$ as $\mathcal{ALN}$- or $\mathcal{ALE}$-TBox (Baader et al., 1999c). Thus, the minimal rewriting decision problem is also NP-hard for $\mathcal{ALN}$ and $\mathcal{ALE}$.

PSPACE-Hardness for $\mathcal{ALC}$

The following lemma yields a reduction of subsumption in $\mathcal{ALC}$ to the minimal rewriting decision problem for $\mathcal{ALC}$. Since subsumption in $\mathcal{ALC}$ is PSPACE-complete (Schmidt-Schauss and Smolka, 1991), this implies PSPACE-hardness for the minimal rewriting decision problem for $\mathcal{ALC}$.

Lemma 3 *Let C, D be two $\mathcal{ALC}$-concept descriptions, and A, P_1, P_2 three different concept names not occurring in C, D. Then $C \sqsubseteq D$ iff there exists a minimal rewriting of size ≤ 1 of the $\mathcal{ALC}$-concept description $P_1 \sqcap P_2 \sqcap C$ using the TBox $\mathcal{T} := \{A \doteq P_1 \sqcap P_2 \sqcap C \sqcap D\}$.*

Proof: First, assume that $C \sqsubseteq D$. This implies $C \equiv C \sqcap D$ and thus $P_1 \sqcap P_2 \sqcap C \equiv P_1 \sqcap P_2 \sqcap C \sqcap D$. Hence, A is a rewriting of size ≤ 1 of $P_1 \sqcap P_2 \sqcap C$ w.r.t. $\mathcal{T}$.

Conversely, let E be a rewriting of size ≤ 1 of $P_1 \sqcap P_2 \sqcap C$ w.r.t. $\mathcal{T}$. We distinguish several cases.

1. $E = A$: Then $P_1 \sqcap P_2 \sqcap C \equiv P_1 \sqcap P_2 \sqcap C \sqcap D$. Since P_1 and P_2 do not occur in C and D, it is easy to show (Baader et al., 1999c) that the above equivalence implies $C \equiv C \sqcap D$, and thus $C \sqcap D$.

2. $E = \bot$: Then $P_1 \sqcap P_2 \sqcap C \equiv \bot$. Since P_1, P_2 are primitive names not occurring in C, we obtain $C \equiv \bot$, and thus $C \sqsubseteq D$.

3. $E = \top$: Then $P_1 \sqcap P_2 \sqcap C \equiv \top$ in contradiction to $P_1 \sqcap P_2 \sqsubset \top$.

4. $E = Q$ for a concept name Q distinct from A: For $Q \in \{P_1, P_2\}$, let w.l.o.g. $Q = P_1$. Then $P_1 \equiv P_1 \sqcap P_2 \sqcap C$. This implies $P_1 \sqsubseteq P_1 \sqcap P_2 \sqcap C$ and hence $P_1 \sqsubseteq P_2$ in contradiction to the fact that P_1 and P_2 are different primitive names. Finally, assume $Q \notin \{A, P_1, P_2\}$. Then $Q \equiv P_1 \sqcap P_2 \sqcap C$ implies $Q \sqsubseteq P_1 \sqcap P_2 \sqcap C$, and hence $Q \sqsubseteq P_1$ in contradiction to the fact that Q and P_1 are different primitive names.

All other cases where $|E| \leq 1$ (e.g., $E = \neg E'$ with $|E'| \leq 1$; $E = \forall r.E'/\exists r.E'$ with $|E'| = 0$; ...) can be treated analogously (see (Baader et al., 1999c) for details). □

This reduction of subsumption to the minimal rewriting decision problem also works for sublanguages of $\mathcal{ALC}$ (if they allow for conjunction) as well as for extensions of $\mathcal{ALC}$ known from the literature. This shows that, for all such DLs, the minimal rewriting decision problem is at least as hard as the subsumption problem. Note that this yields an alternative proof of NP-hardness of the minimal rewriting decision problem for $\mathcal{ALE}$, but not for $\mathcal{FL}_0$ and $\mathcal{ALN}$ (since subsumption is polynomial for these languages).

A general upper bound

The following simple algorithm decides whether there exists a rewriting of C using $\mathcal{T}$ of size $\leq \kappa$ in nondeterministic polynomial time, using an oracle for deciding equivalence modulo TBox: First, nondeterministically compute a concept description E of size $\leq \kappa$; then test whether $E \equiv_{\mathcal{T}} C$.

Note that testing $E \equiv_{\mathcal{T}} C$ is a special case of the general equivalence problem modulo TBox: C does *not* contain defined names. In fact, we have shown that this *restricted equivalence problem* is less complex than the general problem for the DLs $\mathcal{FL}_0$ and $\mathcal{ALN}$ (see (Baader et al., 1999c) for details).

The complexity results for the minimal rewriting decision problem for the DLs under consideration are sum-

TBox	unfolded	not unfolded
$\mathcal{FL}_0$	NP-complete	NP-complete
$\mathcal{ALN}$	NP-complete	in Σ_2^p, NP-hard
$\mathcal{ALE}$	NP-complete	in PSPACE, NP-hard
$\mathcal{ALC}$	PSPACE-complete	PSPACE-complete

Table 2: Summary of the complexity results.

marized in Table 2. The upper bounds are obtained from

- the simple algorithm described above,
- two new complexity results for the restricted equivalence problem for $\mathcal{FL}_0$ (in P) and $\mathcal{ALN}$ (in Δ_2^p), and
- known complexity results for the equivalence problem modulo TBox for $\mathcal{ALE}$ and $\mathcal{ALC}$ (in PSPACE) (Donini et al., 1992; Lutz, 1999).

It should be noted that there are two independent sources of complexity for the minimal rewriting problem. On the one hand, we have to decide equivalence modulo TBox in order to test whether a computed concept description is a rewriting. On the other hand, in order to compute a minimal rewriting, we have to solve an optimization problem. Since the restricted equivalence problem for $\mathcal{FL}_0$ can be decided in polynomial time, the hardness result for $\mathcal{FL}_0$ implies that this optimization problem is hard, independently of the complexity of the equivalence problem.

5 The minimal rewriting computation problem

Whereas the previous section was concerned with deciding whether there exists a (minimal) rewriting within a given size bound, this section considers the problem of actually computing (minimal) rewritings. Due to lack of space, we restrict our attention to $\mathcal{ALE}$, but all notions and results can easily be adapted to $\mathcal{ALN}$ (Baader et al., 1999c).

For a given instance $(C, \mathcal{T})$ of the minimal rewriting *computation problem*, one is interested in either computing (1) *one* minimal rewriting of C using $\mathcal{T}$, or (2) *all* minimal rewritings of C using $\mathcal{T}$.

The hardness results of the previous section imply that computing one minimal rewriting is already a hard problem. In addition, the following simple example shows that the number of minimal rewritings of a concept description C using a TBox $\mathcal{T}$ can be exponential in the size of C and $\mathcal{T}$. This example works for all the four DLs considered in the previous section.

Input: An $\mathcal{ALE}$-concept description C in $\forall$-normal form and an $\mathcal{ALE}$-TBox $\mathcal{T}$.
Algorithm:
Compute an extension C^* of C.
Compute a reduction $\widehat{C}$ of C^* w.r.t. $\mathcal{T}$.
Return $\widehat{C}$.

Figure 1: The rewriting algorithm for $\mathcal{ALE}$.

For a nonnegative integer n, let

$$C_n := P_1 \sqcap \ldots \sqcap P_n \quad \text{and}$$
$$\mathcal{T}_n := \{A_i \doteq P_i \mid 1 \leq i \leq n\}.$$

For each vector $\mathbf{i} = (i_1, \ldots, i_n) \in \{0,1\}^n$, we define

$$E_{\mathbf{i}} := \prod_{1 \leq j \leq n, i_j = 0} P_j \sqcap \prod_{1 \leq j \leq n, i_j = 1} A_j.$$

Obviously, for all $\mathbf{i} \in \{0,1\}^n$, $E_{\mathbf{i}}$ is a rewriting of C_n of size $|E_{\mathbf{i}}| = n = |C_n|$. Furthermore, it is easy to see that there does not exist a smaller rewriting of C_n using $\mathcal{T}_n$. Hence, there exists an exponential number of different minimal rewritings of C_n using $\mathcal{T}_n$.

A *naïve algorithm* for computing one minimal rewriting would enumerate all concept descriptions E of size $k = 1$, then $k = 2$, etc., until a rewriting E_0 of C using $\mathcal{T}$ is encountered. By construction, this rewriting is minimal, and since C is a rewriting of itself, one need not consider sizes larger than $|C|$. If one is interested in computing all minimal rewritings, it remains to enumerate all concept descriptions of size $|E_0|$, and test for each of them whether they are equivalent to C modulo $\mathcal{T}$.

Obviously, this naïve algorithm is very inefficient. Its main drawback is that it is not source-oriented: the candidate rewritings are computed without using the input C. The main contribution of this paper is a nondeterministic rewriting algorithm that computes rewritings by directly modifying the input concept C. More precisely, the algorithm will work on the *$\forall$-normal form* of the input concept, i.e., the normal form obtained from C by exhaustively applying the rule $\forall r.E \sqcap \forall r.F \longrightarrow \forall r.(E \sqcap F)$. This normal form can be computed in polynomial time.

The idea underlying the improved algorithm depicted in Figure 1 is to split the computation of a rewriting E into two steps. First, an extension of C w.r.t. $\mathcal{T}$ is computed.

Definition 4 (Extension) *Let C be an $\mathcal{ALE}$-concept description and $\mathcal{T}$ an $\mathcal{ALE}$-TBox. The concept description C^* is an* extension of C w.r.t. $\mathcal{T}$ *iff $C^* \equiv_{\mathcal{T}} C$ and C^* can be obtained from C by conjoining defined names at some positions in C.*

In the second step, a so-called *reduction of C^* w.r.t. $\mathcal{T}$* is computed, i.e., a concept description $\widehat{C}$ that is (i) equivalent to C^* modulo $\mathcal{T}$, and (ii) obtained from C^* by eliminating all the redundancies in C^*.

The main technical problem to be solved is to give an appropriate formal definition of reduction, and to show how reductions can be computed. Before we go into such detail, we (1) give an example illustrating the rewriting algorithm of Figure 1; and (2) explain what this algorithm actually computes.

(1) As an example, consider the $\mathcal{ACE}$-concept description

$$C = P \sqcap Q \sqcap \forall r.P \sqcap \exists r.(P \sqcap \exists r.Q) \sqcap \exists r.(P \sqcap \forall r.(Q \sqcap \neg Q)),$$

and the $\mathcal{ACE}$-TBox $\mathcal{T} = \{ A_1 \doteq \exists r.Q,\ A_2 \doteq P \sqcap \forall r.P,\ A_3 \doteq \forall r.P \}$. The concept description

$$\begin{aligned} C^* \ = \ & A_2 \sqcap P \sqcap Q \sqcap \forall r.P \sqcap \\ & \exists r.(A_1 \sqcap P \sqcap \exists r.Q) \sqcap \exists r.(P \sqcap \forall r.(Q \sqcap \neg Q)) \end{aligned}$$

is an extension of C. A reduction of C^* can be obtained by eliminating

- P and $\forall r.P$ on the top-level of C^*, because they are redundant w.r.t. A_2;
- P in both of the existential restrictions on the top-level of C^*, because it is redundant due to the value restriction $\forall r.P$;
- the existential restriction $\exists r.Q$, because it is redundant w.r.t. A_1;

and replacing $Q \sqcap \neg Q$ by $\bot$, since $\bot$ is the minimal inconsistent concept description. The resulting concept description $\widehat{C} = A_2 \sqcap Q \sqcap \exists r.A_1 \sqcap \exists r.\forall r.\bot$ is equivalent to C modulo $\mathcal{T}$, i.e., $\widehat{C}$ is a rewriting of C using $\mathcal{T}$. Furthermore, it is easy to see that $\widehat{C}$ is a minimal rewriting of C using $\mathcal{T}$.

(2) Obviously, there may exist exponentially many essentially different (i.e., not equivalent w.r.t. the empty TBox) extensions of C, and we can show (Baader et al., 1999c) that, for $\mathcal{ACE}$, each extension may have exponentially many essentially different reductions (for $\mathcal{ACN}$, reductions are unique). Thus, the algorithm of Figure 1 should be viewed as a nondeterministic algorithm (with an oracle for the equivalence problem). We will show that it is correct in the following sense:

Theorem 5 *1. Every possible output of the algorithm in Figure 1 is a rewriting of the input concept description C using the input TBox $\mathcal{T}$.*

2. The set of all computed rewritings contains all minimal rewritings of C using $\mathcal{T}$ (modulo associativity, commutativity and idempotence of conjunction, and the equivalence $C \sqcap \top \equiv C$).

If we compute just one extension and then one reduction of this extension, then we have a deterministic and polynomial-time algorithm (with an oracle for equivalence) for computing one rewriting; however, the computed rewriting need not be minimal. Nevertheless, this opens the way for a heuristic approach to computing "small" (rather than minimal) rewritings (see Section 6). We can also show the following (Baader et al., 1999c): if we compute all extensions and then just one reduction of each extension, then the set of all rewritings computed this way always contains at least one minimal rewriting.

Reduction of $\mathcal{ACE}$-concept descriptions

For the sake of simplicity, we assume the set of role names N_R to be the singleton $\{r\}$. However, the definitions and results can easily be generalized to arbitrary sets of role names (Baader et al., 1999c).

In order to formalize the notion of a reduction for $\mathcal{ACE}$, we need the following notation.

Definition 6 (Subdescription) *Let $\mathcal{T}$ be an $\mathcal{ACE}$-TBox and C an $\mathcal{ACE}$-concept description that may contain defined names from $\mathcal{T}$. The $\mathcal{ACE}$-concept description $\widehat{C}$ is a* subdescription of C w.r.t. $\mathcal{T}$ *iff* (i) *$\widehat{C} = C$; or* (ii) *$\widehat{C} = \bot$; or* (iii) *$\widehat{C}$ is obtained from C by removing some (negated)* primitive *names, value restrictions, or existential restrictions on the top-level of C, and for all remaining value/existential restrictions $\forall r.D/\exists r.D$ replacing D by a subdescription $\widehat{D}$ of D.*

In the above example, the concept description $\widehat{C}$ is a subdescription of C^*, whereas the concept description $Q \sqcap \exists r.A_1 \sqcap \exists r.\forall r.\bot$ is not since we do not allow for removing defined names in C (unless they occur within value or existential restrictions that are removed as a whole).

Definition 7 (Reduction w.r.t. $\mathcal{T}$) *Let $\mathcal{T}$ be an $\mathcal{ACE}$-TBox and C an $\mathcal{ACE}$-concept description in $\forall$-normal form that may contain defined names from $\mathcal{T}$. An $\mathcal{ACE}$-concept description $\widehat{C}$ is called* reduction of C w.r.t. $\mathcal{T}$ *iff $\widehat{C}$ is a minimal subdescription of C that is equivalent to C modulo $\mathcal{T}$.*

Disallowing the removal of defined names in the definition of the notion "subdescription" makes sense since

Input: An $\mathcal{ALE}$-concept description C in $\forall$-normal form, an $\mathcal{ALE}$-TBox $\mathcal{T}$, and an $\mathcal{ALE}$-concept description F.

Algorithm: reduce$(C, \mathcal{T}, F)$
If $C \sqcap F \equiv_{\mathcal{T}} \bot$, then $\widehat{C} := \bot$;
Otherwise,
Let $\{A_1, \ldots, A_m\} := def(C)$;
Let $\{Q_1, \ldots, Q_\ell\} := prim(C) \setminus prim(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$;
If $val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} val_r(C)$
then $D^r := \top$
else $D^r :=$ reduce$(val_r(C), \mathcal{T}, val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)))$;
Let $\mathcal{D}$ be a subset of the set $\mathcal{C} := \{\text{reduce}(C_j, \mathcal{T}, val_r(C) \sqcap val(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))) \mid C_j \in exr_r(C)\}$ such that

1. there do not exist $D_1, D_2 \in \mathcal{D}$, $D_1 \neq D_2$, with $val_r(C) \sqcap val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqcap D_1 \sqsubseteq_{\mathcal{T}} val_r(C) \sqcap val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqcap D_2$,
2. there does not exist $D \in \mathcal{D}$ with $F \sqcap A_1 \sqcap \ldots \sqcap A_m \sqcap \forall r.val_r(C) \sqsubseteq_{\mathcal{T}} \exists r.D$,
3. for each $C_i \in exr_r(C)$, there exists $D \in \mathcal{D}$ with $\exists r.D \sqcap \forall r.val_r(C) \sqcap F \sqcap A_1 \sqcap \ldots \sqcap A_m \sqsubseteq_{\mathcal{T}} \exists r.C_i$; or $F \sqcap A_1 \sqcap \ldots \sqcap A_m \sqcap \forall r.val_r(C) \sqsubseteq_{\mathcal{T}} \exists r.C_i$, and
4. the size $\sum_{D \in \mathcal{D}}(|D| + 1)$ of the set is minimal among the sizes of all subsets of $\mathcal{C}$ satisfying (1)–(3);

Define $\widehat{C} := Q_1 \sqcap \ldots \sqcap Q_\ell \sqcap A_1 \sqcap \ldots \sqcap A_m \sqcap \forall r.D^r \sqcap \prod_{D \in \mathcal{D}} \exists r.D$,
where the value restriction $\forall r.D^r$ is omitted if $D^r = \top$;
Return $\widehat{C}$.

Figure 2: The reduction algorithm for $\mathcal{ALE}$.

the reduction step is always applied after the extension step. It is possible that removal of defined names could yield a smaller rewriting, but this rewriting is obtained when considering the extension where these names have not been added in the first place. Allowing the removal of defined names would thus only increase the amount of nondeterminism without creating additional rewritings.

In the sequel, we describe an algorithm that computes a reduction of C in deterministic polynomial time (using an oracle for deciding equivalence modulo $\mathcal{T}$). The set of all reductions of C can be computed in exponential time.

Intuitively, a reduction $\widehat{C}$ of an $\mathcal{ALE}$-concept C in $\forall$-normal form is computed in a top-down manner. If $C \equiv_{\mathcal{T}} \bot$, then $\widehat{C} := \bot$. Otherwise, let $\forall r.C'$ be the (unique!) value restriction and $A_1 \sqcap \ldots \sqcap A_n$ the conjunction of the defined names on the top-level of C. Basically, $\widehat{C}$ is obtained from C as follows:

1. Remove the (negated) primitive concept Q occurring on the top-level of C, if $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq_{\mathcal{T}} Q$.
2. Remove $\exists r.C_1$ occurring on the top-level of C, if (a) $A_1 \sqcap \ldots \sqcap A_n \sqcap \forall r.C' \sqsubseteq_{\mathcal{T}} \exists r.C_1$, or (b) there is another existential restriction $\exists r.C_2$ on the top-level of C such that $A_1 \sqcap \ldots \sqcap A_n \sqcap \forall r.C' \sqcap \exists r.C_2 \sqsubseteq_{\mathcal{T}} \exists r.C_1$.
3. Remove $\forall r.C'$ if $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq_{\mathcal{T}} \forall r.C'$.
4. Finally, all concept descriptions D occurring in the remaining value and existential restrictions are reduced recursively.

The formal specification of the reduction algorithm is more complex than the intuitive description given above mainly for two reasons. First, in (2b) it could be the case that the subsumption relation also holds if the rôles of $\exists r.C_1$ and $\exists r.C_2$ are exchanged. In this case, one has a choice of which concept to remove. If the (recursive) reduction of C_1 and C_2 yields descriptions of different size, then we remove the existential restriction for the concept with the larger reduction. If, however, the reductions are of equal size, then we must make a (don't know) nondeterministic choice between removing the one or the other.

Second, in (4) we cannot really reduce the descriptions D without considering the context in which they occur. The reduction of these concepts must take into account the concept C' as well as all concepts D' occurring in value restrictions of the form $\forall r.D'$ on the top-level of the defining concepts for $A_1, \ldots, A_n$. For instance, consider our example from above, where the removal of P within the existential restrictions on the top-level of C^* was justified by the presence of $\forall r.P$ on the top-level of C^*. Since we want to apply the reduction algorithm recursively, we need a third input parameter to take care of the context. To be more precise, the reduction algorithm described in Figure 2 computes a reduction of an $\mathcal{ALE}$-concept description C w.r.t. an $\mathcal{ALE}$-TBox $\mathcal{T}$ and an $\mathcal{ALE}$-concept description F. A *reduction of C w.r.t. $\mathcal{T}$ and F* is a minimal subdescription $\widehat{C}$ of C such that $C \sqcap F \equiv_{\mathcal{T}} \widehat{C} \sqcap F$.

The formal specification of the reduction algorithm in Figure 2 is based on the following notations. Let $\mathcal{T}$ be an $\mathcal{ALE}$-TBox and C an $\mathcal{ALE}$-concept description that may contain defined names from $\mathcal{T}$. The *unfolded concept description* $\mathcal{T}(C)$ is defined as the concept description obtained from C by exhaustively substituting defined names in C by their defining concepts in $\mathcal{T}$.[1] The set of all defined names occurring on the top-level of C is denoted by $def(C)$, and the set of all (negated) primitive names occurring on the top-level of C is denoted by $prim(C)$. For an $\mathcal{ALE}$-concept description C and a role name r,

- $val_r(C)$ denotes the concept description occurring in the unique value restriction on the top-level of the $\forall$-normal form of C, where $val_r(C) := \top$ if there is no such value restriction; and
- $exr_r(C)$ denotes the set $\{C_1, \ldots, C_n\}$ of concept descriptions occurring in existential restrictions of the form $\exists r.C_i$ on the top-level of C.

The following lemma states soundness and completeness of the reduction algorithm of Figure 2.

Lemma 8 *(Baader et al., 1999c) Each output $\widehat{C}$ obtained from* reduce$(C, \mathcal{T}, F)$ *is a reduction of C w.r.t. $\mathcal{T}$ and F. Conversely, for each reduction E of C w.r.t. $\mathcal{T}$ and F, there exists an output $\widehat{C}$ of* reduce$(C, \mathcal{T}, F)$ *that is equal to E.*

Consequently, reduce$(C, \mathcal{T}, \top)$ produces all reductions of C w.r.t. $\mathcal{T}$.

[1] Note that $\mathcal{T}(C)$ is well-defined due to our assumptions on TBoxes. However, just as for unfolding TBoxes, this step may lead to an exponential blow-up.

Proof of Theorem 5

The first item in the statement of Theorem 5 is a direct consequence of the definition of extensions and reductions. In order to prove the second item, let E be a minimal rewriting of C using $\mathcal{T}$. The main point is now that we can define an extension C^* of C induced by E such that E is a subdescription of C^*.

Intuitively, C^* can be obtained from C as follows:

1. conjoin to C all defined names occurring on the top-level of E;
2. if there exists a value restriction $\forall r.E'$ on the top-level of E, then there also exists a value restriction $\forall r.C'$ on the top-level of C (otherwise, C would not be equivalent to E modulo $\mathcal{T}$): substitute C' by the recursively defined extension of C' induced by E';
3. for each existential restriction $\exists r.E_i$ on the top-level of E, there exists a corresponding existential restriction $\exists r.C_i$ on the top-level of C such that $C_i \sqcap val_r(C) \equiv_{\mathcal{T}} E_i \sqcap val_r(C)$ (otherwise, C would not be equivalent to E modulo $\mathcal{T}$): substitute C_i by the recursively defined extension of C_i induced by E_i.

In the formal definition of C^*, we must, just as for the reduction algorithm, take into account the context in which a concept description occurs. To this purpose, we extend the notion "extension w.r.t. $\mathcal{T}$" to "extension w.r.t. $\mathcal{T}$ and F": C^* is an *extension of C w.r.t. $\mathcal{T}$ and F* iff C^* is obtained from C by conjoining defined names from $\mathcal{T}$ at any position in C such that $C^* \sqcap F \equiv_{\mathcal{T}} C \sqcap F$. Furthermore, we need the notion "reduced w.r.t. $\mathcal{T}$ and F": a concept description E is called *reduced w.r.t. $\mathcal{T}$ and F* if E is a reduction w.r.t. $\mathcal{T}$ and F of itself.

The recursive definition of an extension C^* of C induced by E w.r.t. $\mathcal{T}$ and F is depicted in Figure 3. This definition makes sense since it can be shown (Baader et al., 1999c) that there always exists a permutation of $exr_r(C)$ of the desired form.

In order to complete the proof of the second part of Theorem 5, we need the following lemma.

Lemma 9 *(Baader et al., 1999c) Let $\mathcal{T}$ be an $\mathcal{ALE}$-TBox, C, F, E $\mathcal{ALE}$-concept descriptions such that C is in $\forall$-normal form and does not contain defined names, E is reduced w.r.t. $\mathcal{T}$ and F, and $E \sqcap F \equiv_{\mathcal{T}} C \sqcap F$. If C^* is the concept description defined in Figure 3, then*

1. *C^* is an extension of C w.r.t. $\mathcal{T}$ and F, and*

Given: An $\mathcal{ACE}$-TBox $\mathcal{T}$, and $\mathcal{ACE}$-concept description C, F, E, where
- C is in $\forall$-normal form and does not contain defined names,
- E is reduced w.r.t. $\mathcal{T}$ and F, and $C \sqcap F \equiv_{\mathcal{T}} E \sqcap F$.

Recursive definition of the extension C^* of C w.r.t. $\mathcal{T}$ and F induced by E:
If $E \sqcap F \equiv_{\mathcal{T}} \bot$, then $C^* := C$;
Otherwise,
Let $\{Q_1, \ldots, Q_k\} := prim(C)$;
Let $\{A_1, \ldots, A_n\} := def(E)$;
Let D^r be the recursively defined extension of $val_r(C)$ w.r.t. $\mathcal{T}$ and $val_r(F)$ induced by $val_r(E)$;
Let $exr_r(C) = \{C_1, \ldots, C_m\}$ and $exr_r(E) = \{E_1, \ldots, E_\ell\}$;
Let $\{j_1, \ldots, j_m\}$ be a permutation of $\{1, \ldots, m\}$ such that, for all $1 \leq i \leq \ell$,

$$C_{j_i} \sqcap val_r(C \sqcap F) \equiv_{\mathcal{T}} E_i \sqcap val_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n));$$

For $1 \leq i \leq \ell$, let $C^*_{j_i}$ be the recursively defined extension of C_{j_i} w.r.t. $\mathcal{T}$ and $val_r(C \sqcap F)$ induced by E_i;
Then C^* is defined by

$$C^* := Q_1 \sqcap \ldots \sqcap Q_k \sqcap A_1 \sqcap \ldots \sqcap A_n \sqcap \forall r.D^r \sqcap \prod_{1 \leq i \leq \ell} \exists r.C^*_{j_i} \sqcap \prod_{\ell+1 \leq i \leq m} \exists r.C_{j_i},$$

where the value restriction $\forall r.D^r$ is omitted if there does not exist a value restriction on the top-level of C.

Figure 3: The recursive definition of extensions w.r.t. $\mathcal{T}$ and F induced by E.

2. *E is a subdescription of C^*.*

Now, let E be a minimal rewriting of C using $\mathcal{T}$. Then E is reduced w.r.t. $\mathcal{T}$ and $\top$ since otherwise E would not be minimal. Let C^* be the extension of C w.r.t. $\mathcal{T}$ and $\top$ induced by E. By Lemma 9, we know that E is a subdescription of C^*. Thus, minimality of E implies that E is a reduction of C^*, and hence E is contained in the set of all rewritings computed by the algorithm (by Lemma 8).

Complexity of the minimal rewriting computation problem

Using the improved rewriting algorithm for $\mathcal{ACE}$ described in Figure 1, we can show the following complexity results.

Proposition 10 *1. One minimal rewriting of C using $\mathcal{T}$ can be computed using polynomial space.*

2. The set of all *minimal rewritings of C using $\mathcal{T}$ can be computed in exponential time.*

Proof: Each extension of C is polynomial (modulo idempotence) in the size of C and $\mathcal{T}$. Furthermore, there are "only" exponentially many (essentially different) extensions of C. Since equivalence modulo TBox in $\mathcal{ACE}$ can be decided in PSPACE (Lutz, 1999), the set of all extensions can be enumerated using polynomial space. For each extension C^*, the reductions $\widehat{C}$ can again be enumerated in polynomial space. Thus, if we are interested in just one minimal rewriting, it is sufficient always to store the smallest rewriting encountered so far. Hence, we can compute one minimal rewriting of C using polynomial space. Since the number of minimal rewritings may be exponential, the set of all minimal rewritings can only be computed in exponential time. □

6 A heuristic rewriting algorithm

In this section, we present an algorithm that computes a small, but not necessarily minimal, rewriting of an $\mathcal{ACE}$-concept description C using an $\mathcal{ACE}$-TBox $\mathcal{T}$ in *deterministic polynomial time* using an oracle for deciding equivalence modulo $\mathcal{T}$. The idea underlying the algorithm can be described as follows. Instead of first computing an extension of C and then the reduction of this extension, we interleave these two steps in a single pass through the concept. Both, for the extension and the reduction, we employ a *greedy heuristics*. To be more precise, the concept description C is processed recursively. In each recursion step, we build a local extension by conjoining to the top level of C the set $\{A_1, \ldots, A_n\}$ of *all* minimal (w.r.t. $\sqsubseteq_{\mathcal{T}}$) defined names in $\mathcal{T}$ subsuming C. Then we remove *all* (negated) primitive names, value restrictions, and exis-

tential restrictions on the top-level of C that are redundant w.r.t. $A_1, \ldots, A_n$, and the context in which they occur, i.e., the value restrictions obtained from previous recursion steps. Finally, the concept descriptions in the remaining value and existential restrictions are rewritten recursively. Like the reduction algorithm, the heuristic rewriting algorithm thus takes as inputs the concept C to be rewritten, the underlying TBox $\mathcal{T}$, and a concept description F describing the context C has to be considered in.

The formal specification of the rewriting algorithm requires an additional notation. For an $\mathcal{ACE}$-concept description C that may contain defined names, $\mathcal{T}^*(C)$ denotes the concept description obtained from C by exhaustively substituting defined names *on the top-level of* C by their defining concepts from the underlying TBox $\mathcal{T}$. In contrast to $\mathcal{T}(C)$, the size of $\mathcal{T}^*(C)$ is always polynomial in the size of C and $\mathcal{T}$. The following lemma states the correctness and the complexity of the heuristic rewriting algorithm for $\mathcal{ACE}$ depicted in Figure 4.

Lemma 11 *(Baader et al., 1999c) Let $\mathcal{T}$ be an $\mathcal{ACE}$-TBox, C, F $\mathcal{ACE}$-concept descriptions without defined names, and let $\widehat{C}$ be the result of* rewrite$(C, \mathcal{T}, F)$.

1. $\widehat{C} \sqcap F \equiv_{\mathcal{T}} C \sqcap F$.
2. *$\widehat{C}$ is computed in deterministic polynomial time using an oracle for deciding subsumption modulo TBox in $\mathcal{ACE}$.*

The following example shows that the rewriting computed by the heuristic algorithm need not be minimal. For a nonnegative integer $n > 2$, we consider the $\mathcal{ACE}$-concept description $C_n = \forall r.(P_1 \sqcap \ldots \sqcap P_n)$ and the $\mathcal{ACE}$-TBox

$$\begin{aligned} \mathcal{T}_n \;:=\; & \{\, A_i \doteq \forall r.P_i \mid 1 \le i \le n\} \cup \\ & \{A_{n+1} \doteq P_1 \sqcap \ldots \sqcap P_n\}. \end{aligned}$$

The heuristic rewriting algorithm of Figure 4 produces the rewriting $\widehat{C}_n := A_1 \sqcap \ldots \sqcap A_n$ of size n. The unique minimal rewriting of C_n using $\mathcal{T}$ is $E_n := \forall r.A_{n+1}$, which is of size 2. Hence, this example even shows that the difference between the size of the rewriting produced by the heuristic algorithm and the size of the minimal rewritings can become arbitrarily large.

The reason why the heuristic algorithm does not find the minimal rewriting in this example is that it introduces too many defined names on the top level. These names allow for the removal of all the value restrictions on the top level, which makes it impossible to recognize that at a lower level a more promising extension could have been found.

In principle, this is the only reason for the heuristic algorithm not to find a minimal rewriting. In order to characterize the difference between the rewriting computed by the heuristic algorithm and the minimal rewritings, we need the notion of a quasi-subdescription. There are two differences between a subdescription and a quasi-subdescription: on the one hand, in a quasi-subdescription, we do not allow for substituting a concept description by $\bot$. On the other hand, we allow for conjoining defined names at some positions in C.

Definition 12 (Quasi-subdescription) *Let $\mathcal{T}$ be an $\mathcal{ACE}$-TBox and C an $\mathcal{ACE}$-concept description that may contain defined names from $\mathcal{T}$. The $\mathcal{ACE}$-concept description $\widehat{C}$ is a* quasi-subdescription of C w.r.t. $\mathcal{T}$ *iff (i) $\widehat{C} = C$, or (ii) $\widehat{C}$ is obtained from C by removing some (negated) primitive names, conjoining some defined names, removing some value restrictions or existential restrictions, and for all remaining value/existential restrictions $\forall r.D/\exists r.D$ replacing D by a quasi-subdescription $\widehat{D}$ of D.*

Lemma 13 *(Baader et al., 1999c) Let $\mathcal{T}$ be an $\mathcal{ACE}$-TBox, C an $\mathcal{ACE}$-concept description not containing defined names, and E, F $\mathcal{ACE}$-concept descriptions such that E is reduced w.r.t. $\mathcal{T}$ and F and $C \sqcap F \equiv_{\mathcal{T}} E \sqcap F$. The result $\widehat{C}$ of* rewrite$(C, \mathcal{T}, F)$ *is a quasi-subdescription of E.*

If E is a minimal rewriting of C using $\mathcal{T}$, then E is reduced w.r.t. $\mathcal{T}$ and $\top$, and $C \sqcap \top \equiv_{\mathcal{T}} E \sqcap \top$. Thus, Lemma 13 implies that the rewriting $\widehat{C} :=$ rewrite$(C, \mathcal{T}, \top)$ produced by the heuristic algorithm is a quasi-subdescription of E.

Intuitively, if we view concept descriptions as trees where edges are due to existential and value restrictions, and nodes are labeled with (negated) concept names, then the above result can be interpreted as follows. The rewriting $\widehat{C}$ produced by the heuristic algorithm may have a tree structure that is smaller than the one of the minimal rewriting E. The labels of the nodes in the tree corresponding to $\widehat{C}$ may contain less (negated) primitive names, but more defined names.

First experimental results

In order to study the usefulness of our minimal rewriting approach, we have implemented a prototype of the rewriting algorithm depicted in Figure 4. First results obtained in our process engineering application are encouraging: for a TBox with about 65 defined and 55 primitive names, 128 source descriptions of size about 800 (obtained as results of the lcs computation) were

Input: An $\mathcal{ALE}$-concept description C in $\forall$-normal form, an $\mathcal{ALE}$-TBox $\mathcal{T}$, and an $\mathcal{ALE}$-concept description F.

Algorithm: rewrite$(C, \mathcal{T}, F)$
If $C \sqcap F \equiv_{\mathcal{T}} \bot$, then $\widehat{C} := \bot$;
If $F \sqsubseteq_{\mathcal{T}} C$, then $\widehat{C} := \top$;
Otherwise,
Let $\{A_1, \ldots, A_n\}$ be the set of all minimal defined names A_i with $C \sqcap F \sqsubseteq_{\mathcal{T}} A_i$;
Let $\{Q_1, \ldots, Q_\ell\} := prim(C) \setminus prim(\mathcal{T}^*(F \sqcap A_1 \sqcap \ldots \sqcap A_n))$;
Let $D^r := \text{rewrite}(val_r(C), \mathcal{T}, val_r(\mathcal{T}^*(F \sqcap A_1 \sqcap \ldots \sqcap A_n)))$;
Let $\{D_1, \ldots, D_m\} := exr_r(C)$ and $\mathcal{D}^r := \{D_1, \ldots, D_m\}$;
For $i = 1, \ldots, m$ do
if (1) there exists $D \in \mathcal{D}^r \setminus \{D_i\}$ with $D \sqcap val_r(C \sqcap \mathcal{T}^*(F)) \sqsubseteq D_i$, or
(2) $A_1 \sqcap \ldots \sqcap A_n \sqcap val_r(C) \sqcap F \sqsubseteq \exists r.D_i$
then $\mathcal{D}^r := \mathcal{D}^r \setminus \{D_i\}$;
Define $\widehat{C} := Q_1 \sqcap \ldots \sqcap Q_\ell \sqcap A_1 \sqcap \ldots \sqcap A_n \sqcap \forall r.D^r \sqcap \bigsqcap_{D \in \mathcal{D}^r} \exists r.\text{rewrite}(D, \mathcal{T}, val_r(C \sqcap \mathcal{T}^*(F)))$,
where $\forall r.D^r$ is omitted if $D^r = \top$;
Return $\widehat{C}$.

Figure 4: A rewriting algorithm for $\mathcal{ALE}$ using a greedy heuristics.

rewritten into descriptions of size about 10.

For each of these rewritings, the set of defined names computed in each recursion step, i.e., the set $\{A_1, \ldots, A_n\}$ in Figure 4, had size one, i.e., there existed just one (minimal) defined name subsuming $C \sqcap F$. Thus, the negative effect (illustrated by the above example) that too many defined names were conjoined did not occur in our experiments. For the future, we are planning a more thorough empirical evaluation, also comparing the heuristic algorithm with one that actually computes (all) minimal rewritings.

7 Related and future work

In this paper, we have restricted our attention to the minimal rewriting problem. There are, however, also other interesting instances of the general rewriting framework introduced in Section 3.

Rewriting queries using views

The problem of *rewriting queries using views* in DLs, as considered in (Beeri et al., 1997), is one such instance. As source and TBox-DL, that paper considers the language $\mathcal{ACN}$ and its extension $\mathcal{ACCNR}$,[2] i.e., $\mathcal{L}_s = \mathcal{L}_t = \mathcal{ACN}$ or $\mathcal{L}_s = \mathcal{L}_t = \mathcal{ACCNR}$, and as destination DL $\mathcal{L}_d = \{\sqcap, \sqcup\}$. The rewritings to be computed are maximally contained rewritings, i.e., the relation ρ is subsumption $\sqsubseteq$, and the ordering $\preceq$ is inverse subsumption $\sqsupseteq$. More precisely, (Beeri et al., 1997) is concerned with *total* rewritings, i.e., the rewriting E should no longer contain primitive names. In our framework, total rewritings can be taken into account by modifying the optimality ordering $\preceq$ as follows: $E \preceq E'$ iff (a) E does not contain primitive names and E' contains primitive names, or (b) E and E' do not contain defined names and $E \sqsupseteq E'$. If there exists at least one total rewriting E of C using $\mathcal{T}$, then each minimal (w.r.t. the modified ordering $\preceq$) rewriting of C is total.

Section 3 of (Beeri et al., 1997) contains the following two results:

- For $\mathcal{L}_s = \mathcal{L}_t = \mathcal{ACCNR}$ and $\mathcal{L}_d = \{\sqcap, \sqcup\}$, a maximally contained total rewriting is computable. Using the subsumption algorithm for $\mathcal{ACCNR}$, this can also be used to decide whether there exists a total rewriting equivalent to the input concept C.
- If $\mathcal{ACCNR}$ is replaced by $\mathcal{ACN}$, then one can compute a maximally contained total rewriting in exponential time, and existence of a total rewriting equivalent to C can also be decided in exponential time.

It should be noted that, in the conclusion of (Beeri et al., 1997), the authors state that, for $\mathcal{ACN}$, a maximally contained rewriting can be computed in polynomial time; however, the actual complexity bound given in (Beeri et al., 1997), Theorem 3.2, only yields an exponential time bound. This coincides with our

[2] In addition to the constructors in $\mathcal{ACC}$, $\mathcal{ACCNR}$ allows for number restrictions and *role conjunction* $(r_1 \sqcap r_2)$.

complexity results given in Section 4.

Translation of concept descriptions

Another interesting instance of the framework, which we intend to investigate in the future, is the *translation of concept descriptions* from one DL into another, i.e., the instance where (i) $\mathcal{L}_s$ and $\mathcal{L}_d$ are *different* DLs; (ii) the TBox is assumed to be empty; and (iii) the binary relation ρ is given as $\equiv$, $\sqsubseteq$, or $\sqsupseteq$. By trying to rewrite an $\mathcal{L}_s$-concept C into an *equivalent* $\mathcal{L}_d$-concept E, one can find out whether C is expressible in $\mathcal{L}_d$. In many cases, such an exact rewriting may not exist. In this case, one can try to approximate C by an $\mathcal{L}_d$-concept from above (below), i.e., find a minimal (maximal) concept description E in $\mathcal{L}_d$ such that $C \sqsubseteq E$ ($E \sqsubseteq C$). An inference service that can compute such rewritings could, for example, support the transfer of knowledge bases between different systems.

References

Baader, F. and Küsters, R. (1998). Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ACN}$-concept descriptions. In *Proceedings of the 22nd Annual German Conference on Artificial Intelligence (KI'98)*, volume 1504 of *Lecture Notes in Computer Science*, Springer Verlag.

Baader, F. and Küsters, R. (2000). Matching in description logics with existential restrictions. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*. Morgan Kaufmann.

Baader, F., Küsters, R., Borgida, A., and McGuinness, D. (1999a). Matching in description logics. *Journal of Logic and Computation*, 9(3).

Baader, F., Küsters, R., and Molitor, R. (1999b). Computing least common subsumers in description logics with existential restrictions. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence 1999 (IJCAI'99)*, Morgan Kaufmann.

Baader, F., Küsters, R., and Molitor, R. (1999c). Rewriting concepts using terminologies – revisited. LTCS-Report 99-12, LuFG Theoretical Computer Science, RWTH Aachen, Germany. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

Baader, F. and Sattler, U. (1996). Knowledge representation in process engineering. In *Proceedings of the 1996 International Workshop on Description Logic (DL'96)*, AAAI Press.

Beeri, C., Levy, A. Y., and Rousset, M.-C. (1997). Rewriting queries using views in description logics. In *PODS '97. Proceedings of the Sixteenth ACM SIG-SIGMOD-SIGART Symposium on Principles of Database Systems*, ACM Press.

Borgida, A. and McGuinness, D. L. (1996). Asking queries about frames. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, Morgan Kaufmann.

Cohen, W., Borgida, A., and Hirsh, H. (1992). Computing least common subsumers in description logics. In *Proceedings of the 10th National Conference on Artificial Intelligence*, MIT Press.

Cohen, W. and Hirsh, H. (1994). Learning the CLASSIC description logic: Theoretical and experimental results. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR'94)*, Morgan Kaufmann.

Donini, F., Lenzerini, M., Nardi, D., Hollunder, B., Nutt, W., and Spaccamela, A. (1992). The complexity of existential quantification in concept languages. *Artificial Intelligence*, 53.

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco.

Lutz, C. (1999). Complexity of terminological reasoning revisited. In *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, volume 1705 of *Lecture Notes in Artificial Intelligence*, Springer Verlag.

Nebel, B. (1990a). *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence*. Springer Verlag.

Nebel, B. (1990b). Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43(2).

Sattler, U. (1998). *Terminological knowledge representation systems in a process engineering application*. PhD thesis, RWTH Aachen.

Schmidt-Schauss, M. and Smolka, G. (1991). Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1).

Diagnosis

Formulating diagnostic problem solving using an action language with narratives and sensing

Chitta Baral
Department of CSE
Arizona State University
Tempe, AZ 85287
chitta@asu.edu

Sheila McIlraith
Knowledge Systems Lab
Stanford University
Stanford, CA 94305
sam@ksl.stanford.edu

Tran Cao Son
CS Department
Univ. of Texas at El Paso
El Paso, Texas 79968
tson@cs.utep.edu

Abstract

Given a system and unexpected observations about the system, a diagnosis is often viewed as a fault assignment to the various components of the system that is consistent with (or that explains) the observations. If the observations occur over time, and if we allow the occurrence of (deliberate) actions and (exogenous) events, then the traditional notion of a candidate diagnosis must be modified to consider the *possible occurrence of actions* and events that could account for the unexpected system behavior.

In the presence of multiple candidate diagnoses, we may need to perform actions and observe their impact on the system, to be able to narrow the list of possible diagnoses, and possibly even initiate some repair. A plan that guarantees such narrowing will be referred to as a *diagnostic plan*, and if this plan also guarantees that at the end of the execution of the plan, the system has no faults then we refer to it as a *repair plan*.

Since actions and narrative play a central role in diagnostic problem solving, we characterize diagnosis, diagnostic planning and repair with respect to the existing action language $\mathcal{L}$, extended to include static constraints, sensing actions, and the notion of observable fluents. This language is used to provide a uniform account of diagnostic problem solving.

1 Introduction

Consider the following narrative involving diagnosis.

> John gets up in the morning. He turns on the switch of his lamp, and reads the morning newspaper. He then turns off the switch and does other things before going to work. After he gets home from work, he enters his room and turns on the switch of his lamp again. This time, *the lamp does not turn on.* John thinks that maybe either the bulb is broken, or the switch of the lamp is broken, or the power cord is broken, or there is no power at the outlet. He does nothing about it and goes to his bathroom and turns on the light switch, observing that even that light does not turn on. He thinks perhaps there is no power at home, but then he notices that his electric clock is working, so he figures that there is power in at least part of his home. Now he is worried and goes to his garage to check his fuse box and finds that one of the fuses is blown. He replaces that fuse and comes back to his room. He turns on his lamp switch and voila it works.

This narrative illustrates the process of diagnostic problem solving. In particular it illustrates that diagnostic problem solving must involve reasoning about the evolution of a dynamical system. Triggered by an observation of system behavior that is inconsistent with expected behavior – in this case, the fact that when John turned on the lamp it did not emit light, diagnostic problem solving involves:

- generating candidate diagnoses based on an incomplete history of events that have occurred and observations that have been made.
- in the event of multiple candidate diagnoses, performing actions to enable observations that will discriminate candidate diagnoses. The selection of a particular actions is often biased towards confirming the

most likely diagnosis, or the one that is easiest to test.
- generating plans (possibly with conditionals and sensing actions) to perform these discriminatory observations.
- updating the space of diagnoses in the face of changes in the state of the world, and in the face of new observations.

The long-term objective of our work is to develop a knowledge representation and reasoning capability that emulates diagnostic problem solving processes such as John's. Following [McI97b], we argue that such a comprehensive account of diagnostic problem solving must involve reasoning about action and change. In this paper we augment and extend the work of [McI97a, McI98, McI97b] in several important ways. The main contributions of this paper are:
- We define diagnosis with respect to a narrative.
- We define the notion of diagnostic and repair planning, within a language that integrates sensing actions and world-altering actions. Thus, we are able to distinguish between changes in the state of the world, and changes in an agent's state of knowledge.
- In support of this endeavor, we extend the action language $\mathcal{L}$ to support static causal laws, sensing actions and the notion of observable fluents. $\mathcal{L}$ was originally developed to support narratives (e.g., [MS94, Pin94]).

None of the above issues have been explored either in the model-based diagnosis literature or in the reasoning about action literature. Also notable is that unlike most other accounts of diagnosis, our account allows nondeterministic effects of actions. Finally, our work is distinguished from most previous work in defining diagnosis in terms of a diagnostic model, rather than in terms of failing components and/or actions sequences.

The rest of the paper is organized as follows. In Section 2 we give an overview of the language $\mathcal{L}$ and how to add static causal laws to it. In Section 3 we use the extended language to define when we may need to do a diagnosis and what a diagnosis is with respect to a narrative. In Section 4 we further extend our action language to allow sensing actions and to accommodate the distinction between an observable fluent and a unobservable fluent. We then use this language to define the notion of a conditional plan, and the related notions of diagnostic and repair planning. Finally, in Section 5 we summarize and discuss related work.

2 Specifying narrative in $\mathcal{L}$

The propositional language $\mathcal{L}$ was developed in [BGP97, BGP98] to specify narratives and to reason with them. In this paper, we will describe the main aspects of the language $\mathcal{L}$ by dividing it into three components: a domain description language $\mathcal{L}_D$, a language to specify observations $\mathcal{L}_O$, and a query language $\mathcal{L}_Q$. In Section 4.1, we extend our language further with sensing actions, and observables.

2.1 $\mathcal{L}_D$: The domain description language

The alphabet of $\mathcal{L}_D$ – a language that closely follows the language $\mathcal{AC}$ from [Tur97] – comprises two nonempty disjoint sets of symbols: the set of fluents **F**, and the set of actions, **A**. A *fluent literal* (or *literal*) is a fluent or a fluent preceded by $\neg$. A *fluent formula* is a propositional formula constructed from literals. Propositions in $\mathcal{L}_D$ are of the following forms:

$$a \textbf{ causes } \varphi \textbf{ if } \psi \quad (1)$$

$$\varphi \textbf{ if } \psi \quad (2)$$

$$\textbf{impossible } a \textbf{ if } \psi \quad (3)$$

where a is an action, and φ, and ψ are fluent formulas.

Propositions of the form (1) describe the direct effects of actions on the world and are called *dynamic causal laws*. Propositions of the form (2), called *static causal laws*, describe causal relation between fluents in a world. Propositions of the form (3), called *executability conditions*, state when actions are not executable.

A *domain description* D is a set of propositions in $\mathcal{L}_D$.

The main difference between $\mathcal{L}_D$ and the action description part of $\mathcal{L}$ [BGP97, BGP98] is the presence of static causal laws in $\mathcal{L}_D$, which are critical for representing the behavior of the device being diagnosed.

A domain description given in $\mathcal{L}_D$ defines a transition function from actions and states to a set of states. (Recall, actions may be nondeterministic.) Intuitively, given an action, a and a state, s the transition function $\Phi(a, s)$ defines the set of states that may be reached after executing the action a in state s. If $\Phi(a, s)$ is an empty set it means that a is not executable in s. We now formally define this transition function.

Let D be a domain description in the language of $\mathcal{L}_D$. An *interpretation* I of the fluents in $\mathcal{L}_D$ is a maximal consistent set of fluent literals of $\mathcal{L}_D$. A fluent f is said to be true (resp. false) in I iff $f \in I$ (resp. $\neg f \in I$). The truth value of a fluent formula in I is defined recursively over the propositional connective in the usual way. For example, $f \wedge q$ is true in I iff f is true in I and q is true in I. We say that φ holds in I (or I satisfies φ), denoted by $I \models \varphi$, if φ is true in I.

A set of formulas from $\mathcal{L}_D$ is *logically closed* if it is closed under propositional logic (wrt $\mathcal{L}_D$).

Let V be a set of formulas and K be a set of static causal laws of the form φ **if** ψ. We say that V is closed under K if for every rule φ **if** ψ in K, if ψ belongs to V then so does φ. By $Cn(V \cup K)$ we denote[1] the least logically closed set of formulas from $\mathcal{L}_D$ that contains V and is also closed under K.

A *state* of D is an interpretation that is closed under the set of static causal laws of D.

An action a is *prohibited (not executable)* in a state s if there exists an executability condition of the form

impossible a **if** φ

in D such that φ holds in s.

The *effect of an action* a in a state s is the set of formulas $e_a(s) = \{\varphi \mid D$ contains a law a **causes** φ **if** ψ and ψ holds in $s\}$.

Given the domain description D containing a set of static causal laws R, we formally define $\Phi(a, s)$, the set of states that may be reached by executing a in s as follows.

1. If a is not prohibited (i.e., executable) in s, then

$$\Phi(a, s) = \{s' \mid Cn(s') = Cn((s \cap s') \cup e_a(s) \cup R)\};$$

2. If a is prohibited (i.e., not executable) in s, then $\Phi(a, s)$ is $\emptyset$.

The intuition behind the above formulation is as follows. The direct effects (due to the dynamic causal laws) of an action a in a state s are given by $e_a(s)$, and all formulas in $e_a(s)$ must hold in any resulting state. In addition, the static causal laws (R) dictate additional formulas that must hold in the resulting state. While the resulting state should satisfy these formulas, it must also be otherwise closed to s. These three aspects are captured by the definition above. For additional explanation and motivation behind the above definition please see [Tur97].

2.2 $\mathcal{L}_O$: The observation language

We assume the existence of a set of situation constants S which contains two special situation constants s_0 and s_c denoting the initial situation and the current situation, respectively. Note that *situations* written as s (possibly with subscripts) are different from *states* which are written as s (possibly with subscripts). As with the situation calculus, the ontology of our language differentiates between a situation, which is a history of the actions from the initial situation, and a state, which is the truth value of fluents at a particular situation.

Observations in $\mathcal{L}_O$ are propositions of the following forms:

$$\varphi \textbf{ at } s \tag{4}$$
$$\alpha \textbf{ between } s_1, s_2 \tag{5}$$
$$\alpha \textbf{ occurs\_at } s \tag{6}$$
$$s_1 \textbf{ precedes } s_2 \tag{7}$$

where φ is a fluent formula, α is a (possibly empty) sequence of actions, and s, s_1, s_2 are situation constants which differ from s_c. (Since the world can be changed without the agent's knowledge, we do not allow the agent to have observations about s_c.)

Observations of the forms (4) and (7) are called *fluent facts* and *precedence facts*, respectively. Observations of the forms (5) and (6) are referred to as *occurrence facts*. These two types of observations are different in that (5) states exactly what happened between two situations s_1 and s_2, whereas (6) only says what occurred in the situation s.

2.3 Narratives

A *narrative* is a pair (D, Γ) where D is a domain description and Γ is a set of observations of the form (4)-(7).

Observations are interpreted with respect to a domain description. While a domain description defines a transition function that characterize what states *may* be reached when an action is executed in a state, a narrative consisting of a domain description together with a set of observations defines the possible situation histories of the system. This characterization is achieved by two functions, Σ and Ψ. While Σ maps situation constants to action sequences, Ψ picks one among the various transitions given by $\Phi(a, s)$ and maps action sequences to a unique state with the condition that $\Psi(\alpha \circ a) \in \Phi(a, \Psi(\alpha))$.

More formally, let (D, Γ) be a narrative. A *causal interpretation* of (D, Γ) is a partial function from action sequences to interpretations of $Lang(\mathbf{F})$, whose

[1] Note that a fluent formula φ can be equivalently represented as a static causal law φ **if** *true*.

domain is nonempty and prefix-closed[2]. By $Dom(\Psi)$ we denote the domain of a causal interpretation Ψ. Notice that $[] \in Dom(\Psi)$ for every causal interpretation Ψ, where [] is the empty sequence of actions.

A *causal model* of D is a causal interpretation Ψ such that:

(i) $\Psi([])$ is a state of D; and

(ii) for every $\alpha \circ a \in Dom(\Psi)$, $\Psi(\alpha \circ a) \in \Phi(a, \Psi(\alpha))$.

A *situation assignment* of $\mathbf{S}$ with respect to D is a mapping Σ from $\mathbf{S}$ into the set of action sequences of D that satisfy the following properties:

(i) $\Sigma(s_0) = []$;

(ii) for every $s \in \mathbf{S}$, $\Sigma(s)$ is a prefix of $\Sigma(s_c)$.

An *interpretation* M of (D, Γ) is a pair (Ψ, Σ), where Ψ is a causal model of D, Σ is a situation assignment of $\mathbf{S}$, and $\Sigma(s_c)$ belongs to the domain of Ψ. For an interpretation $M = (\Psi, \Sigma)$ of (D, Γ):

(i) α **occurs_at** s is true in M if the sequence $\Sigma(s) \circ \alpha$ is a prefix of $\Sigma(s_c)$;

(ii) α **between** s_1, s_2 is true in M if $\Sigma(s_1) \circ \alpha = \Sigma(s_2)$;

(iii) φ **at** s is true in M if φ holds in $\Psi(\Sigma(s))$;

(iv) s_1 **precedes** s_2 is true in M if $\Sigma(s_1)$ is a prefix of $\Sigma(s_2)$.

An interpretation $M = (\Psi, \Sigma)$ is a *model* of a narrative (D, Γ) if:

(i) facts in Γ are true in M;

(ii) there is no other interpretation $M' = (\Psi, \Sigma')$ such that M' satisfies condition i) above and $\Sigma'(s_c)$ is a subsequence of $\Sigma(s_c)$.

Observe that these models are minimal in the sense that they exclude extraneous actions.

A narrative is *consistent* if it has a model. Otherwise, it is *inconsistent.*

[2]A set X of action sequences is prefix-closed if for every sequence $\alpha \in X$, every prefix of α is also in X.

2.4 $\mathcal{L}_Q$: The query language

Queries in $\mathcal{L}_Q$ are of the following form:

$$\varphi \textbf{ after } \alpha \textbf{ at } s \tag{8}$$

When α in (8) is an empty sequence of actions, and s is the current situation s_c, we often use the notation **currently** φ as a simplification of (8).

A query of the form φ **after** α **at** s is true in a model $M = (\Psi, \Sigma)$ if φ is true in $\Psi(\Sigma(s) \circ \alpha)$.

A query q is entailed by a narrative (D, Γ), denoted by $(D, \Gamma) \models q$, if q is true in every model of (D, Γ).

3 Diagnosis wrt narratives

We are now ready to formulate the notion of diagnosis with respect to a narrative. The representation of the system to be diagnosed comprises static causal laws that describes the behavior of the system itself, as well as the description of the effects of actions on system state, and observations about action occurrences and fluent values over the evolution of the system We follow the diagnosis literature (e.g., [DMR92]) and assume that the system is composed of a distinguished set of components that can malfunction. Associated with each component c, is the distinguished fluent $ab(c)$, denoting that the component c is abnormal or broken. Also associated with each component is the distinguished fluent $break(c)$, a wildcard action which may be used to explain unexpected observations about $ab(c)$. Note that the representation of the system is likely to contain other actions and static causal laws that affect the truth of $ab(c)$. Building on the established diagnosis notation:

Definition 1 (System)
A *system Sys* is a tuple $(SD, COMPS, OBS)$ where

$COMPS = \{c_1, \ldots, c_n\}$ is a finite set of components.

SD is a domain description characterizing the behavior of the system, and augmented with dynamic laws of the form $break(c)$ **causes** $ab(c)$, for each component c in $COMPS$.

Given SD, by SD_{ab}, we denote the subset of SD consisting of static causal laws of the form "ψ **if** φ" and dynamic laws of the form "a **causes** ψ **if** φ", where ψ contains $ab(c)$ for some component c.

OBS is a collection of observations starting from the situation s_1, and the precedence fact

s_0 **precedes** s_1. Specification of fluent facts at s_0 are not included in OBS.

In our formulation of diagnosis, we make the assumption that there is an initial situation in the history where all components are operating normally[3]. This is achieved by adding the set $OK_0 = \{\neg ab(c)$ **at** $s_0 \mid c \in COMPS\}$ to our observations.

Example 1 Consider a slight variation of the story in our introduction. Assume that the only breakable component in the domain is the *bulb*. Furthermore, assume that John observed that the light is off *immediately* after he turned on the lamp when coming back from work. The story can then be described by a system description $Sys_0 = (SD_0, \{bulb\}, OBS_0)$, where

SD_0 :

$$\begin{array}{c} turn\_on \textbf{ causes } light\_on \textbf{ if } \neg ab(bulb) \\ turn\_off \textbf{ causes } \neg light\_on \\ \neg light\_on \textbf{ if } ab(bulb) \\ break(bulb) \textbf{ causes } ab(bulb) \\ \textbf{impossible } break(bulb) \textbf{ if } ab(bulb) \end{array}$$

OBS_0 :

$turn\_on$ **occurs_at** s_1	s_0 **precedes** s_1
$turn\_off$ **occurs_at** s_2	s_1 **precedes** s_2
$turn\_on$ **between** s_3, s_4	s_2 **precedes** s_3
$\neg light\_on$ **at** s_1	s_3 **precedes** s_4
$light\_on$ **at** s_2	
$\neg light\_on$ **at** s_3	
$\neg light\_on$ **at** s_4	

OK_0 : $\neg ab(bulb)$ **at** s_0. □

Intuitively, we say a system needs a diagnosis, if the following assumptions are inconsistent with the observations (i) all components are initially fine, and (ii) no action that can break a component occurs. To define diagnosis, we assume that all components were initially operating normally, and we try to conjecture minimal action occurrences to account for the observations. Since the semantics of $\mathcal{L}$ minimizes action occurrences, all we need to do is to consider the various models of the narrative and extract our diagnosis from each.

[3]Hence, our formulation of diagnosis can be alternately referred to as 'big-bang diagnosis'. We can slightly modify it to define *incremental diagnosis*, when we already know that a set $X \subseteq COMP$ is abnormal, and we want to figure out if some additional components have malfunctioned by having $OK_0 = \{ab(c)$ **at** $s_0 \mid c \in X\} \cup \{\neg ab(c)$ **at** $s_0 \mid c \in COMP \setminus X\}$.

Definition 2 (Necessity of Diagnosis) We say a system $Sys = (SD, COMPS, OBS)$ needs a diagnosis if the narrative $(SD \setminus SD_{ab}, OBS \cup OK_0)$ does not have a model.

Note that the notion of a system needing a diagnosis is not meant to capture the notion that there is some fault in the system. It is a much weaker notion. We now establish the notion of a diagnosis in terms of a diagnostic model.

Definition 3 (Diagnostic Model) Let $Sys = (SD, COMPS, OBS)$ be a system that needs a diagnosis. We say M is a diagnostic model of Sys if M is a model of the narrative $(SD, OBS \cup OK_0)$.

We can now extract information about any particular situation from the diagnostic model. In particular,

Definition 4 (Diagnosis) A diagnosis with respect to situation s is the set of components $\Delta \in COMPS$ such that there exists a model $M = (\Psi, \Sigma)$ of the narrative $(SD, OBS \cup OK_0)$ and $\Delta = \{c \mid ab(c)$ **at** s holds in $M\}$. We refer to a diagnosis with respect to s_c as a *current fluent diagnosis*. We say a diagnosis Δ (wrt a situation s) is *minimal* if there exists no diagnosis Δ' (wrt s) such that $\Delta' \subset \Delta$.

Example 2 (Continuation of Example 1) Consider the system $Sys_0 = (SD_0, \{bulb\}, OBS_0)$, from Example 1, with $SD_{ab} = \{break(bulb)$ **causes** $ab(bulb)\}$.

Let narrative $N_0' = (SD_0 \setminus SD_{ab}, OBS_0 \cup OK_0)$. Due to the proposition "$\neg light\_on$ **if** $ab(bulb)$", $SD_0 \setminus SD_{ab}$ has only three distinct states: $s_0 = \emptyset$, $s_1 = \{light\_on\}$, and $s_2 = \{ab(bulb)\}$.

The transition function of $SD_0 \setminus SD_{ab}$ is given by

$$\begin{array}{ll} \Phi(turn\_on, s_0) = \{s_1\} & \Phi(turn\_off, s_0) = \{s_0\} \\ \Phi(turn\_on, s_1) = \{s_1\} & \Phi(turn\_off, s_1) = \{s_0\} \\ \Phi(turn\_on, s_2) = \{s_2\} & \Phi(turn\_off, s_2) = \{s_2\} \end{array}$$

We now prove that N_0' is inconsistent. Assume the contrary, N_0' has a model (Σ, Ψ). Because of $OBS_0 \cup OK_0$, we conclude that $\Psi([]) = s_0$. Let $\Sigma(s_3) = \alpha$, where α is an action sequence. By the definition of a model of a narrative, we have that $\Sigma(s_4) = \alpha \circ turn\_on$. As there is no action in $SD_0 \setminus SD_{ab}$ whose effect is $ab(bulb)$, we conclude that $ab(bulb) \notin \Psi(\alpha)$. This implies that $light\_on \in \Psi(\alpha \circ turn\_on)$, i.e., $light\_on$ must hold in s_4. This contradicts the observation "$\neg light\_on$ **at** s_4", i.e., N_0' is inconsistent.

Narrative N_0' is inconsistent, and hence, Sys_0 needs a diagnosis. We compute the diaguosis as follows.

The narrative $N_0 = (SD_0, OBS_0 \cup OK_0)$, has one model $M = (\Psi, \Sigma)$ where $\Psi([]) = s_0$ and

$\Sigma(s_0) = \Sigma_1(s_1) = []$, $\Sigma(s_2) = turn\_on$,
$\Sigma(s_3) = turn\_on \circ turn\_off \circ break(bulb)$,
$\Sigma(s_4) = \Sigma(s_c) = turn\_on \circ turn\_off \circ break(bulb) \circ turn\_on$.

We can easily verify that $ab(bulb)$ **at** s_c is true in M. Hence a current diagnosis for Sys_0 is $\Delta = \{bulb\}$. Moreover, it is easy to check that Δ is also a minimal current diagnosis for Sys_0. □

3.1 Explanation vs diagnosis

Often the observations in a narrative can be *explained* by the sequence of actions (possibly exogenous) that have occurred. Unfortunately, this is not true in all cases because incomplete knowledge of the initial situation, and/or non-deterministic actions can lead to uncertainty in the outcome of a sequence of actions.

The definition of a diagnostic model in the previous section uses a *consistency* criterion to account for the observations. That is, the narrative (SD, Γ), where Γ comprises the sequence of action occurrences and initial situation (including OK_0) dictated by the diagnostic model, do not necessarily *entail* OBS. They are merely consistent with OBS. Here we define the notion of an explanatory diagnostic model, which has the stronger criterion that (SD, Γ) must entail OBS.

Definition 5 (Explanatory Diagnostic Model) Suppose $M = (\Psi, \Sigma)$ is a diagnostic model of (SD, $COMPS$, OBS), where

- $actions(M)$ is the set of occurrence facts and precedence facts of the forms (5), (6), and (7), (i.e., facts of the forms α **between** s_1, s_2, α **occurs_at** s_1, s_1 **precedes** s_2) that are true in M; and

- initial(M) is the set of fluent facts of the form f **at** s_0 that are true in M, including OK_0.

Then M is an *explanatory diagnostic model* iff

$$(SD,\ actions(M) \cup initial(M)) \models OBS$$

Following in this spirit, it is straightforward to define the notion of an explanatory diagnosis, a set of action occurrences that entails the observations.

4 Diagnostic and repair planning

The diagnostic process discussed in the previous section will generate a set of candidate diagnoses, however diagnosis is only the first step in dealing with an errant system. In most cases we will attempt to discriminate these diagnoses with the objective of identifying a unique diagnosis and/or reducing our space of candidate diagnoses to a point where a repair plan can be conceived. We are operating under the assumption that we cannot directly observe the state of abnormality of the various components of the system. Nevertheless, we can make other observations about the system, add them to OBS, and then refine our diagnoses.

In general, the fluents in the system are of two kinds: *observable* and *unobservable*. A simple *generic observation*[4] leads the agent to know the value of the observable fluents. By knowing the relationship between the observable and unobservable fluents, and the values of the observable fluents, we can sometimes deduce the values of unobservable fluents. We can also use direct sensing actions to sometimes determine their value. Given a set of candidate diagnoses, we can execute a plan – perhaps including some sensing actions and conditional branches – and make the generic observations to obtain additional information that will help reduce the space of possible diagnoses. Such plans are distinguished in that they can have knowledge goals in addition to goals relating to the state of the world. We refer to plans that attempt to reduce our space of diagnoses as *diagnostic plans*. A diagnostic plan that includes some repair is called a *repair plan*.

4.1 Adding sensing and observables to $\mathcal{L}$

In order to define the notion of a diagnostic plan, we must first augment $\mathcal{L}_D$ and $\mathcal{L}_Q$ to incorporate sensing actions and observable and unobservable fluents. In this section, we briefly describe these augmentations. The resulting theories are called $\mathcal{L}_{DS}$ and $\mathcal{L}_{QS}$, respectively. ($\mathcal{L}_O = \mathcal{L}_{OS}$.)

• We allow *knowledge producing laws* of the following form in $\mathcal{L}_{DS}$:

$$\textbf{a determines } f \qquad (9)$$

where a is an action and f is a fluent. A law of this form tells us that after a is executed, the value of the

[4]Here we distinguish between generic observations and sensing actions. We assume that the agent is constantly performing 'generic observations' and thus knows the truth value of the observable fluents at all times. In contrast, sensing actions require the agent's effort.

fluent f will be known. An action occurring in a knowledge producing law is called a *sensing action*.

• With the addition of sensing actions, we need to distinguish between a state of the world and the state of the agent's knowledge about the world. The later will be referred to as a *combined state* (or *c-state*) and will be represented by a pair of the form $\langle s, \mathcal{S}\rangle$, where s is a state (representing the real state of the world) and $\mathcal{S}$ is a set of states (representing the set of states an agent thinks it may be in).

• We extend the transition function Φ to also map pairs of actions and c-states into sets of c-states.

1. for any c-state $\langle s, \mathcal{S}\rangle$ and non-sensing action a,

   $\Phi(a, \langle s, \mathcal{S}\rangle) = \{\langle s', \mathcal{S}'\rangle \mid s' \in \Phi(a, s)$, and $\mathcal{S}'$ is the set of states in $\Phi(a, \mathcal{S})$ that agree with s' on $\mathbf{F_O}$, the **observable** literals $\}$.

   (Note that if a is not executable in $\langle s, \mathcal{S}\rangle$ then $\Phi(a, \langle s, \mathcal{S}\rangle) = \emptyset$.)

2. for any c-state $\langle s, \mathcal{S}\rangle$ and sensing action a whose knowledge producing laws are
   a determines f_1 ... **a determines** f_m

   (a) if a is executable in $\langle s, \mathcal{S}\rangle$, $\Phi(a, \langle s, \mathcal{S}\rangle) = \{\langle s, \{s' \mid s' \in \mathcal{S}$ such that s and s' agree on the literals from $\mathbf{F_O} \cup \{f_1, \ldots, f_m\}\}\rangle\}$;
   (b) otherwise, $\Phi(a, \langle s, \mathcal{S}\rangle) = \emptyset$.

• In the presence of incomplete information and knowledge producing actions, there may not exist simple plans consisting of sequence of actions and we may need to extend the notion of a plan to allow conditional statements. We refer to such plans as conditional plans (e.g., [Lev96, BS97, BS98]), described below.

• In order to query the system, we specify a query language $\mathcal{L}_{QS}$. A query in $\mathcal{L}_{QS}$ has the form

$$\varphi \textbf{ after } P \textbf{ at } s \tag{10}$$

where φ is a fluent formula and P is a conditional plan as formally defined below.

Definition 6 (Conditional Plan)

1. An empty sequence of action, denoted by [ ], is a conditional plan.

2. If a is an action then a is a conditional plan.

3. If $P_1, \ldots, P_n$ are conditional plans and φ_j's are conjunction of fluent literals (which are mutually exclusive but not necessarily exhaustive) then the following is a conditional plan. (We refer to such a plan to as a *case plan*).

   Case
   $\varphi_1 \rightarrow P_1$
   ...
   $\varphi_n \rightarrow P_n$
   Endcase

4. If P_1 and P_2 are conditional plans then $P_1; P_2$ is a conditional plan.

5. Nothing else is a conditional plan.

In order to define when a narrative entails a query that includes a conditional plan, we need to define an *extended transition function* $\hat{\Phi}$, that maps a pair of a conditional plan and a c-state, into a set of c-states. Intuitively, if $\sigma' \in \hat{\Phi}(P, \sigma)$ then the execution of the plan in the c-state σ may take us to the c-state σ'. Before defining $\hat{\Phi}$, we first define the possible trajectories when P is executed in σ.

Definition 7 Let P be a conditional plan and σ be a c-state. We say a sequence of c-states $\sigma_1, \ldots, \sigma_n$ is a trajectory of P wrt σ if:

1. $P = []$, and $n = 1$, and $\sigma_1 = \sigma$.

2. $P = [a]$, and $n = 2$, and $\sigma_1 = \sigma$ and $\sigma_2 \in \Phi(a, \sigma)$.

3. $P =$ Case
   $\varphi_1 \rightarrow P_1$
   ...
   $\varphi_n \rightarrow P_n$
   Endcase,

   and there exists an i such that φ_i is known to be true in σ and $\sigma_1, \ldots, \sigma_n$ is a trajectory of P_i wrt σ.

4. $P = P_1; P_2$, and $\sigma_1 = \sigma$, and $\sigma_1, \ldots, \sigma_k$ is a trajectory of P_1 wrt σ, and $\sigma_{k+1}, \ldots, \sigma_n$ is a trajectory of P_2 wrt σ_{k+1}.

σ_n is referred to as the resulting c-state of P wrt σ.

Definition 8 Let P be a conditional plan and $\sigma = \langle s, \mathcal{S}\rangle$ be a c-state, $\hat{\Phi}(P, \sigma)$ is now defined as follows:

1. $\hat{\Phi}([], \sigma) = \{\sigma\}$;

2. For an action a, $\hat{\Phi}(a, \sigma) = \Phi(a, \sigma)$;

3. For P = Case
$\varphi_1 \rightarrow P_1$
...
$\varphi_n \rightarrow P_n$
Endcase,

$$\hat{\Phi}(P,\sigma) = \begin{cases} \hat{\Phi}(P_i,\sigma) & \text{if } \varphi_i \text{ is known to be true in } \sigma \\ \emptyset & \text{if there exists no } i \text{ s.t. } \varphi_i \text{ is known to be true in } \sigma \end{cases}$$

4. For $P = P_1; P_2$, where P_1 is a conditional plan and P_2 is a conditional plan,
   - if $\hat{\Phi}(P_1,\sigma) \neq \emptyset$, and for every $\sigma' \in \hat{\Phi}(P_1,\sigma)$, $\hat{\Phi}(P_2,\sigma') \neq \emptyset$, then $\hat{\Phi}(P,\sigma) = \bigcup_{\sigma' \in \hat{\Phi}(P_1,\sigma)} \hat{\Phi}(P_2,\sigma')$; and
   - $\hat{\Phi}(P,\sigma) = \emptyset$ otherwise.

It should be noted that $\hat{\Phi}(P,\sigma)$ is not equal to the set of resulting c-states of P wrt σ. This is because some branches of P may lead to unexecutable actions and hence $\hat{\Phi}(P,\sigma)$ will be empty while there may be several trajectories corresponding to other branches.

Our next goal is to define entailment of queries wrt narratives. Intuitively, since the narrative may not be complete, or have sufficient observations to arrive at a unique model, multiple models may tell us that a situation s may correspond to many different states, only one of which corresponds to s in reality. Thus we have a set of c-states from which we need to verify the correctness of a conditional plan with respect to a goal. More formally,

Definition 9 (Possible State wrt a Situation) Let $N = (D,\Gamma)$ be a narrative. We say s is a possible state corresponding to situation s, if there exists a model (Ψ,Σ) of N such that $\Psi(\Sigma(\mathsf{s})) = s$. We say $\sigma = \langle s,\mathcal{S}\rangle$ is a c-state corresponding to situation s, if s is a possible state corresponding to situation s and $\mathcal{S}$ is the set of all possible states corresponding to s.

A query $q = \varphi$ **after** P **at** s of $\mathcal{L}_{QS}$ is said to be entailed by narrative (D,Γ), i.e. $(D,\Gamma) \models q$, if for every c-state $\langle s,\mathcal{S}\rangle$ corresponding to s, $\hat{\Phi}(P,\langle s,\mathcal{S}\rangle) \neq \emptyset$ and φ is known to be true in every c-state belonging to $\hat{\Phi}(P,\langle s,\mathcal{S}\rangle)$.

4.2 Diagnostic and repair plans

We are now ready to define what a diagnostic plan is. Intuitively, it is a conditional plan, possibly with sensing actions which when executed in the current situation gives sufficient information to reach a unique diagnosis. In addition, we may have certain restrictions, such as that:

- Certain literals are not allowed to change during the execution of the plan. We refer to such literals as *protected literals*. (E.g., to stabilize the leaning tower of Pisa, we may not tear down and rebuilt it.)
- Certain literals are allowed to change during the execution of the plan, but we require that at the end of the execution of the plan, their value be the same as it was before the plan was executed. We refer to such literals as *restored literals*. (E.g., disassembling an engine or a flashlight to diagnose it, but putting it back together afterwards.)
- Certain literals are allowed to change during the execution of the plan, but we require that at the end of the plan, their value be either the same as it was before the plan was executed or be *false*. Such literals will be referred to as *fixable literals*. (This accommodates repair, where *ab* fluents can be made $\neg ab$.)

Definition 10 (Diagnostic Plan) Given $Sys = (SD, COMPS, OBS)$ with a set of protected literals L_P, a set of restored literals L_R, and a set of fixable literals L_F. Let $C \subseteq COMPS$. A conditional plan P is called a *diagnostic plan* for Sys wrt (C, L_P, L_R, L_F), if for every c-state $\sigma = \langle s,\mathcal{S}\rangle$ corresponding to the current situation of Sys, $\hat{\Phi}(P,\sigma) \neq \emptyset$ and for all trajectories of the form $\sigma_1,\ldots,\sigma_n$ of P wrt σ

(i) for every c-state $\langle s',\mathcal{S}'\rangle$ in $\hat{\Phi}(P,\langle s,\mathcal{S}\rangle)$ and $s'' \in \mathcal{S}'$, $s \sim_{AB(C)} s''$ where $AB(C) = \{ab(c) \mid c \in C\}$;

(ii) value of all literals in L_P remain unchanged (wrt the real states) in the trajectory;

(iii) for all literals l in L_R value of l (wrt the real states) in σ_1 and σ_n are the same; and

(iv) for all atoms f in L_F, if f is false in σ_1 then it is false in σ_n. (wrt the real state).

If $C = COMPS$, and the ab-literals are part of L_P, we say that P is a *purely diagnostic plan* for Sys.

If C is a singleton, i.e., $C = \{c\}$ for some $c \in COMPS$, and $ab(c)$ and $\neg ab(c)$ are in L_P, we say that P is a *discriminating diagnostic plan* for c.

Definition 11 (Repair Plan) A diagnostic plan P for Sys wrt (C, L_P, L_R, L_F) is said to be a *repair plan* wrt (C', L_P, L_R, L_F) if (i) $C' \subseteq C \subseteq COMPS$, (ii) ab literals about C' are not in L_P and L_R, and (iii) for every c-state $\sigma = \langle s,\mathcal{S}\rangle$ corresponding to the current situation of Sys, $\hat{\Phi}(P,\sigma) \neq \emptyset$ and for all $c \in C'$, $\neg ab(c)$ is known to be true in all c-states in $\hat{\Phi}(P,\langle s,\mathcal{S}\rangle)$.

Example 3 (Electro-magnetic Door)
Consider an electro-magnetic control door. The door is connected to a RED LED and a YELLOW LED. To enter, an agent needs to put its electro-magnetic card, containing its id-number and password, into the slot connected to the door's controller. The door will open only if the card is valid, the id-number and the password are not corrupted, and the door is not malfunctioning. While the card is in the slot, if it is invalid, the RED LED will be on; and if the id-number or the password is corrupted or the door is defective, the YELLOW LED will be on. The YELLOW LED is on only if the RED LED is not. In this case, pushing the button "message" will print out a message. Reading it, the agent will know whether the door is defective or its card is unreadable.

Our agent, Jack comes to work, and as usual, puts his card into the slot. The door does not open. What is wrong ? The story can be represented by the system $Sys_1 = (SD_1, \{card, door, id\_pwd\}, OBS_1)$ as follows.

The actions of the domain description SD_1 are: *insert_card*, *push_button*, *take_out_card*, look (look at the LEDs), or read_msg (read the message).

The fluents of SD_1 are: *ab(card)*, *ab(door)*, *ab(id_pwd)*, *has_card*, *card_in_slot*, *door_open*, *has_msg*, *red*, and *yellow*, where *red* or *yellow* indicate that the RED/YELLOW LED is on, respectively.

SD_1 comprises the following laws:

• dynamic causal laws: describing the effects of the actions *insert_card*, *push_button*, and *take_out_card*. Inserting the card causes the door to open if the card, the door and the card information are all normal. Further, inserting the card causes the card to be in the slot and not in the possession of the agent. I.e.,

insert_card **causes** *door_open*
if $\neg ab(card), \neg ab(door), \neg ab(id\_pwd)$

insert_card **causes** $\neg has\_card \wedge card\_in\_slot$

Pushing the button results in a message. I.e.,

push_button **causes** *has_msg*.

If the card is in the slot and the agent takes it out, then the agent has possession of the card and the card is not in the slot. I.e.,

take_out_card **causes** $has\_card \wedge \neg card\_in\_slot$
if *card_in_slot*

• static causal laws: expressing the relationship between the status (on/off) of the LEDs. I.e.,

red **if** $ab(card) \wedge card\_in\_slot$
$\neg red$ **if** $yellow \wedge card\_in\_slot$
$yellow$ **if** $(ab(id\_pwd) \vee ab(door)) \wedge \neg red \wedge card\_in\_slot$

• sensing actions: characterizing the knowledge effects of sensing actions. For example, performing the look action causes the agent to know whether the RED and YELLOW LEDs are on or off. They are captured by the following k-propositions:

look **determines** *red*
look **determines** *yellow*
read_msg **determines** $ab(id\_pwd)$
read_msg **determines** $ab(door)$

• executability conditions: characterizing when an action is precluded. I.e.,

impossible *insert_card* **if** $\neg has\_card$
impossible *push_button* **if** $\neg yellow$
impossible read_msg **if** $\neg has\_msg$

• wildcard actions:

$break(card)$ **causes** $ab(card)$
$break(door)$ **causes** $ab(door)$
$break(id\_pwd)$ **causes** $ab(id\_pwd)$

and the set of observations, OBS_1:

$\neg red \wedge \neg yellow \wedge \neg door\_open$ **at** s_1
$has\_card \wedge \neg has\_msg$ **at** s_1
$\neg card\_in\_slot$ **at** s_1
insert_card **between** s_1, s_2
$\neg door\_open$ **at** s_2
s_0 **precedes** s_1
s_1 **precedes** s_2

The first three observations describe the first observable situation, s_1. The fourth observation states that Jack puts his card into the slot, while the fifth states that the door is not open after Jack puts his card into the slot.

Intuitively, when Jack observes that the door does not open as the result of putting his card into the slot, he should realize that at least one of the three components: the card, the door, or the information

on the card is no longer valid. Our diagnostic reasoning systems does likewise. Indeed, the narrative $N_1' = (SD_1 \setminus SD_{ab}, OBS_1 \cup OK_0)$ does not have a model and there are three diagnoses for Sys_1: $\Delta_1 = \{ab(id\_pwd)\}$, $\Delta_2 = \{ab(door)\}$, and $\Delta_3 = \{ab(card)\}$ which correspond to the models M_1, M_2, and M_3 of $N_1 = (SD_1, OBS_1 \cup OK_0)$ defined as follows. $M_1 = (\Psi_1, \Sigma_1)$, $M_2 = (\Psi_2, \Sigma_2)$, and $M_3 = (\Psi_3, \Sigma_3)$, where $\Psi_1([]) = \Psi_2([]) = \Psi_3([]) = s_0$, and

$\Sigma_1(s_0) = []$,
$\Sigma_1(s_1) = break(id\_pwd)$,
$\Sigma_1(s_2) = \Sigma_1(s_c) = break(id\_pwd) \circ insert\_card$,

$\Sigma_2(s_0) = []$,
$\Sigma_2(s_1) = break(door)$,
$\Sigma_2(s_2) = \Sigma_2(s_c) = break(door) \circ insert\_card$,

$\Sigma_3(s_0) = []$,
$\Sigma_3(s_1) = break(card)$,
$\Sigma_3(s_2) = \Sigma_3(s_c) = break(card) \circ insert\_card$.

where $s_0 = \{has\_card\}$,
$\Psi_1(break(id\_pwd)) = \{has\_card, ab(id\_pwd)\}$,
$\Psi_1(break(id\_pwd) \circ insert\_card)$
$= \{card\_in\_slot, ab(id\_pwd), yellow\} = s_1$,

$\Psi_2(break(door)) = \{has\_card, ab(door)\}$,
$\Psi_2(break(door) \circ insert\_card)$
$= \{card\_in\_slot, ab(door), yellow\} = s_2$,

$\Psi_3(break(card)) = \{has\_card, ab(card)\}$,
$\Psi_3(break(card) \circ insert\_card)$
$= \{card\_in\_slot, ab(card), red\} = s_3$.

To narrow the list of the possible diagnoses of the system, Jack can find out the status of the LEDs. If the RED LED is on, he knows for sure that the card is no longer valid. Otherwise, the YELLOW LED must be on. In that case, he can get the message and read it to know if the door is broken or the information on the card is corrupted. This process is captured by the following plan.

$P = look \circ$
  case
    $red \rightarrow []$
    $\neg red \rightarrow$
      case
        $yellow \rightarrow push\_button \circ read\_msg$
      endcase
  endcase

We will now show that P is a diagnostic plan for Sys_1 wrt $(C, L_P, \emptyset, \emptyset)$ where $C = \{id\_pwd, dood, card\}$ and $L_P = \{ab(id\_pwd), ab(dood), ab(card)\}$.

Let $\mathcal{S} = \{s_1, s_2, s_3\}$. There are three possible current situations of Sys_1: $\sigma_1 = \langle s_1, \mathcal{S}\rangle$, $\sigma_2 = \langle s_2, \mathcal{S}\rangle$, and $\sigma_3 = \langle s_3, \mathcal{S}\rangle$. Let $s_i' = s_i \cup \{has\_msg\}$, $i = 1, 2$, then

$\hat{\Phi}(P, \sigma_1) = \hat{\Phi}(push\_button \circ read\_msg, \langle s_1, \{s_1, s_2\}\rangle)$
$= \hat{\Phi}(read\_msg, \langle s_1', \{s_1', s_2'\}\rangle) = \{\langle s_1', \{s_1'\}\rangle\}$;

$\hat{\Phi}(P, \sigma_2) = \hat{\Phi}(push\_button \circ read\_msg, \langle s_2, \{s_1, s_2\}\rangle)$
$= \hat{\Phi}(read\_msg, \langle s_2', \{s_1', s_2'\}\rangle) = \{\langle s_2', \{s_2'\}\rangle\}$;

$\hat{\Phi}(P, \sigma_3) = \{\langle s_3, \{s_3\}\rangle\}$.

The above computations also represent all trajectories of P wrt σ_1, σ_2, and σ_3. Obviously, $\hat{\Phi}(P, \sigma_i) \neq \emptyset$ for $i = 1, 2, 3$. Furthermore, it is easy to check that the values of literals in L_P remain unchanged in all trajectories and in each c-state $\langle s', \mathcal{S}'\rangle$ belonging to $\hat{\Phi}(P, \sigma_i)$ and $s'' \in \mathcal{S}'$, $s' \sim_{AB(C)} s''$. For example, $\langle s_1', \{s_1'\}\rangle$ is the only c-state in $\hat{\Phi}(P, \sigma_1)$, and trivially, $s_1' \sim_{AB(C)} s_1'$. Thus, P is a diagnostic plan for Sys_1 wrt $(C, L_P, \emptyset, \emptyset)$. □

5 Summary and Related Work

In this paper we provided an account of diagnostic problem solving in terms of the action language, $\mathcal{L}$. A prime objective of this work was to characterize diagnostic problem solving with narrative and sensing. $\mathcal{L}$ proved ideal for this task because it already had most of the necessary expressive power. In particular, $\mathcal{L}$ includes narrative, sensing actions, and additionally nondeterministic actions, which are common in diagnostic domains. In this paper, we extended $\mathcal{L}$ by adding static causal laws that are necessary for describing the behavior of the systems we diagnose. We also distinguished notions of observable fluents, and protected, restored and fixable fluents.

The main contributions of this paper, in addition to the supporting language extensions, are the characterization of the diagnosis task as a narrative understanding task, and the definition of diagnosis in terms of a diagnostic model – a particular model of the narrative. We further distinguish between a diagnostic model and the stricter notion of an explanatory diagnostic model. As discussed throughout the paper, diagnostic problem solving is more than just determining a set of candidate diagnoses. In the second half of the paper, we define the notion of a diagnostic plan, and a repair plan – conditional plans that exploit both world-altering actions and sensing actions

with the goal of achieving some diagnostic knowledge or repair objective. These present new contributions to the research on model-based diagnosis and reasoning about action.

We contrast our contributions to related work. In the area of diagnosis of dynamical systems, there has been research both within the control theory community (e.g., [SSLST96]) on the diagnosis of discrete event systems using finite state automata, and within the AI community. Most of this work is fairly recent, and can be differentiated with respect to the expressive power of the language used to model the domain (e.g., propositional/first order, ramifications, nondeterministic actions, concurrent actions, narrative, sensing, probabilities); how the notion of diagnosis is defined (e.g., models, sequences of actions, sets of abnormal components, probabilistic criteria); how observations are expressed; whether diagnosis is active or offline; and what aspects of diagnostic problem solving, beyond diagnosis, are addressed (e.g., diagnostic planning, repair).

Our work was influenced by previous work of McIlraith (e.g., [McI97a, McI98, McI97b]), but extends and builds on aspects of that work in several important ways. [McI97b] argued that a comprehensive account of diagnostic problem solving must involve reasoning about action and change, and provided such an account in a dialect of the situation calculus that included causal ramification constraints, but did not include nondeterministic actions, sensing actions or narrative. Aberrant behavior was assumed to be caused by unobserved exogenous actions. Multiple definitions of diagnosis were provided both in terms of sequences of actions that explained the observations, and designations of normal and abnormal components with respect to a situation. The notion of a diagnostic model was not employed. Most importantly, this account did not exploit narrative for expressing and accounting for observations, consequently the assertion of observations and exogenous actions was much less elegant. [McI97b] also introduced the notion of testing to discriminate hypotheses, and analogues to the ideas of diagnostic and repair planning; however since the dialect of the situation calculus she employed did not include knowledge-producing actions, the important integration of sensing and world-altering actions that was done in this paper, was argued for but was left to future work.

Also of note is the work of Thielscher on a theory of dynamic diagnosis in the fluent calculus [Thi97]. Thielscher characterizes diagnoses in terms of minimally failing components, where his minimization preference criterion is with respect to the abnormalities in the initial state, but can additionally exploit some a prior likelihood. Thielscher does not exploit exogenous actions to account for abnormalities as we do, and does not allow for the occurrences of actions beyond what are observed. Thielscher does not take his work beyond a characterization of diagnosis.

A third important piece of work from the AI community is the work of Cordier, Thiébaux and their coauthors, (e.g., [TCJK96, CT94]). Their work is similar in spirit to ours, viewing the diagnosis task as the determination of an event-history of a system between successive observations. While this work is related, the representation of the domain uses state transition diagrams and is much less expressive and elaboration tolerant than ours. That said, their representation system is sufficiently expressive for the power distribution domain they have been examining, and more recently, their work has focused on the necessary tradeoffs required to address hard computational issues associated with their domain. Cordier and Thiébaux also discuss the notion of repair planning, but without distinguishing between sensing and world-altering actions.

Other notable work on the diagnosis of dynamical systems includes the work of Nayak and Williams on online mode identification for the NASA remote agent system (e.g., [WN96]), the work of Baroni et al. on the diagnosis of large active concurrent systems [BLPZ99], and work on temporal aspects of diagnosis by Brusoni et al. (e.g., [BCTD98]).

In the area of diagnostic and repair planning, Sun and Weld [SW93] proposed a decision-theoretic planner which was invoked by a diagnostic reasoner to plan repair actions. The associated planning language distinguished between information-gathering and state-altering actions, but did not provide for the specification of knowledge or diagnostic goals. Similarly Heckerman et al. [HBR94] have examined the problem of interactively generating repair plans under uncertainty using Bayes nets, a single fault assumption and a myopic lookahead heuristic. Actions are limited to simple observations and component replacement. In contrast Friedrich et al. (e.g., [FN92]) developed a set of greedy algorithms to choose between performing simple observations and repair actions, assuming a most likely diagnosis. They do not limit their system to repair alone but rather generalize their goal to some notion of purpose; purpose does not include specification of diagnostic goals. Finally, and perhaps most notably, Rymon [Rym93] developed a goal-directed diagnostic reasoner and companion planner, called TraumAID 2.0. The primary task of the diagnostic reasoner was to generate goals for the planner and to reason about whether

those goals were satisfied.

While the above work has some of the same goals as our work, none has an explicit notion of knowledge, and hence the integration of sensing and world-altering actions is engineered, rather than treated formally within a logic. In the spirit of such integration, we point to the work of [SL93] and more recently [Lev96] and [DL99] as examples of actions theories with knowledge and sensing; however, to the best of our knowledge ours is the first action theory that allows both sensing and narrative.

References

[BLPZ99] P. Baroni, G. Lamperti, P. Pogliano and M. Zanella Diagnosis of large active systems *Artificial Intelligence*, 110(1):135–183, 1999.

[BGP97] C. Baral, M. Gelfond, and A. Provetti. Representing Actions: Laws, Observations and Hypothesis. *Journal of Logic Programming*, 31(1-3):201–243, 1997.

[BGP98] C. Baral, A. Gabaldon, and A. Provetti. Formalizing Narratives using nested circumscription. *Artificial Intelligence*, 104(1-2):107–164, 1998.

[BS97] C. Baral and T. Son. Regular and special sensing in robot control - relation with action theories. In *Proc. of AAAI 97 Workshop on Robots, Softbots, and Immobots - Theories of Action, Planning and Control*, 1997.

[BS98] C. Baral and T. Son. Relating theories of actions and reactive control. *Electronic Transactions on Artificial Intelligence*, 2(3-4), 1998.

[BCTD98] V. Brusoni, L. Console, P. Terenziani, and D. Theseider Dupré. A spectrum of definitions for temporal model based diagnosis. *Artificial Intelligence*, 102:39–79, 1998.

[CT94] M. Cordier and S. Thiébaux. Event-based diagnosis for evolutive systems. Technical Report 819, IRISA, Cedex, France, 1994.

[DL99] G. DeGiacomo and H. Levesque. Projection using regression and sensors. In *IJCAI 99*, pages 160–165, 1999.

[DMR92] J. de Kleer, A.K. Mackworth and R. Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2–3):197–222, 1992.

[FN92] G. Friedrich and W. Nejdl. Choosing observations and actions in model-based diagnosis/repair systems. In *Proc. of KR 92*, pages 489–498, 1992.

[HBR94] D. Heckerman and J. Breese and K. Rommelse. Troubleshooting under uncertainty. Microsoft Research, Technical Report MSR-TR-94-07, 1994.

[Lev96] H. Levesque. What is planning in the presence of sensing? In *Proc. of AAAI 96*, pages 1139–1146, 1996.

[McI94] S. McIlraith. Generating tests using abduction. In *Proc. of KR 94*, pages 449–460, 1994.

[McI97a] S. McIlraith. Representing actions and state constraints in model-based diagnosis. In *Proc. of AAAI 97*, pages 43–49, 1997.

[McI97b] S. McIlraith. *Towards a formal account of diagnostic problem solving.* PhD thesis, University of Toronto, 1997.

[McI98] S. McIlraith. Explanatory diagnosis: Conjecturing actions to explain observations. In *Proc. of KR 98*, pages 167–177, 1998.

[MS94] R. Miller and M. Shanahan. Narratives in the situation calculus. *Journal of Logic and Computation*, 4(5):513–530, 1994.

[Pin94] J. Pinto. *Temporal Reasoning in the Situation Calculus.* PhD thesis, University of Toronto, Department of Computer Science, 1994.

[Rei87] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[Rym93] R. Rymon. Diagnostic reasoning and planning in exploratory-corrective domains. PhD thesis, University of Pennsylvania, 1993.

[SSLST96] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen and D. Teneketzis. Failure diagnosis using discrete-event models. *IEEE Trans. on Control Systems Technology*, vol. 4, no. 2, pp. 105-124, 1996.

[SL93] R. Scherl and H. Levesque. The frame problem and knowledge producing actions. In *Proc. of AAAI 93*, pages 689–695, 1993.

[SW93] Y. Sun and D. Weld. A framework for model-based repair. In *Proc. of AAAI 93*, pages 182–187, 1993.

[TCJK96] S. Thiébaux, M.-O. Cordier, O. Jehl and J.-P. Krivine. Supply Restoration in Power Distribution Systems – A Case Study in Integrating Model-Based Diagnosis and Repair Planning. In *Proc. of UAI 96*, pages 525–532, 1996.

[Thi97] M. Thielscher. A theory of dynamic diagnosis. *ETAI*, 2(11), 1997. Available electronically at http://www.ep.liu.se/ea/cis/1997/011.

[Tur97] H. Turner. Representing actions in logic programs and default theories. *Journal of Logic Programming*, 31(1-3):245–298, 1997.

[WN96] B. Williams and P. Nayak. A model-based approach to reactive self-configuring systems. In *AAAI 96*, pages 971–978, 1996.

Anytime Diagnostic Reasoning using Approximate Boolean Constraint Propagation

Alan Verberne*
Dept. of AI, Faculty of Sciences
Vrije Universiteit Amsterdam
a.s.verberne@research.kpn.com

Frank van Harmelen
Dept. of AI, Faculty of Sciences
Vrije Universiteit Amsterdam
Frank.van.Harmelen@cs.vu.nl

Annette ten Teije
Dept. of CS
University of Utrecht
annette@cs.uu.nl

Abstract

In contrast with classical reasoning, where a solution is either correct or incorrect, *approximate reasoning* tries to compute solutions which are close to the ideal solution, without necessarily being perfect. Such approximate reasoning can be used to exchange solution quality for computation time, known as anytime reasoning.

In this paper we study approximate versions of *diagnostic reasoning*. Traditionally, diagnostic reasoning is characterised in terms of the logical entailment relation. In this paper we study the effects of replacing the logical entailment relation with an approximate version of the entailment relation, in particular an approximate version of Boolean Constraint Propagation (BCP).

We characterise the cheapest versions of approximate BCP which allows single components and entire systems to be diagnosed correctly. From these upperbounds surprisingly low values follow which are needed to correctly diagnose many of the typical circuit examples from the literature. A particularly interesting property that we discovered is that the point at which approximate diagnosis coincides with classical diagnosis is entirely determined by the nature of the individual components, and not by either the size or the structural complexity of the overall device.

Keywords: Deduction, Diagnosis

*currently at KPN Research

1 MOTIVATION

Approximate reasoning is a form of reasoning that can be used for solving complex problems. For approximate reasoning one defines a quality measure on the output, and the computation then tries to optimize this quality measure. This is in contrast with the conventional notion of a solution, where a solution is either correct or incorrect, with no middle ground. Optimising the quality measure does provide such a middle ground.

Approximate reasoning is interesting for several reasons. First of all, most AI-problems (tasks) are hard problems in term of their complexity measure. Planning, diagnosis and configuration are all examples of AI tasks for which even simple varieties are already intractable (e.g. [Bylander *et al.*, 1991]). Therefore it is necessary to look for cheaper but approximate solutions instead of intractable precise solutions. Secondly, it depends on the particular application of the problem type (e.g. design, diagnosis) whether a precise solution is actually needed or whether an approximate solution suffices. For instance, in diagnostic reasoning there is not always a need for computing precise diagnoses, for example in cases where all possible diagnoses will result in the same repair action (e.g. when all fault-candidates are located on the same computerboard that must be replaced or in the medical domain when all possible diagnoses indicate the same drug to be prescribed) [vanHarmelen & tenTeije, 1995]. As a third reason, it is often not possible in practice to have complete and correct data and knowledge. Examples are missing attribute values in classification, incomplete medical knowledge for performing diagnosis and incomplete requirements for design. So again, an approximate answer should be computed if possible, because an approximate answer is often better than no answer at all.

A particularly interesting form of approximate reason-

ing is *anytime reasoning* [Dean & Boddy, 1988; Boddy & Dean, 1989; Zilberstein & Russell, 1996]. The most important characteristic of anytime reasoning is that with increasing runtime, the quality of the solution increases. Furthermore, the reasoning can be interrupted at any time and will return the best result computed until then. In this paper, we will concentrate on this type of reasoning.

Many types of reasoning in AI such as diagnosis, classification, design and planning are characterised in terms of logical entailment [Reiter, 1987; Green, 1969; Rosenschein, 1981]. A general approach for constructing approximate problem solving behaviour is to use an approximation of the logical entailment for characterising such a problem type and see what conclusions can be drawn using approximate reasoning for a particular problem. This approach was already taken in [tenTeije & vanHarmelen, 1996], where we used the approximate entailment relations defined in [Schaerf & Cadoli, 1995] to characterise approximate diagnostic reasoning. In this paper we consider the sound but incomplete approximation of the logical entailment from [Dalal, 1996] in the context of diagnostic reasoning. The approximation of Dalal is called BCP_k and is based on *boolean constraint propagation*. The parameter k can be increased to improve the quality (and of course also the cost) of this approximate entailment relation as will be explained in section 3.

The general question that we study is *"how can we use BCP_k for approximate diagnostic reasoning?"*. This question boils down to questions such as "how does the quality of the diagnoses improve with time using BCP_k?", "for which k do the BCP_k diagnoses coincide with the classical diagnoses?", "what can be said about the incorrect answers that BCP_k gives?", etc.

The structure of the paper is as follows: to make this paper self-contained we repeat briefly in section 2 the standard definitions of consistency-based diagnosis from the literature. Section 3 introduces the approximate entailment relation we will exploit for diagnostic reasoning. The main results of this paper are discussed in section 4, where we present examples and theorems that characterise the behaviour of the approximate diagnostic reasoning that results from applying BCP_k in the standard definition of diagnosis. Finally, section 6 summarises and concludes.

2 CONSISTENCY-BASED DIAGNOSIS

We briefly repeat the widely accepted definitions from [Reiter, 1987]:

Definition 1 (From [Reiter, 1987])

- A *system* is a pair (SD, COMP) where SD, the *system description* is a set of sentences in first order logic, and COMP the set of system components is a set of constants. The system description uses abnormality predicates $ab(c)$ for every $c \in \text{COMP}$, interpreted to mean that component c does not function normally.
- The *observations* are a set of first order sentences OBS.
- A *diagnosis problem* exists *iff* $\text{SD} \cup \text{OBS} \cup \{\neg ab(c) | c \in \text{COMP}\}$ is inconsistent.
- A *candidate-set* is set $\Delta \subseteq \text{COMP}$.
- A *diagnosis* is a non-empty candidate-set Δ such that $\text{SD} \cup \text{OBS} \cup \{\neg ab(c) | c \in \text{COMP} \setminus \Delta\}$ is consistent.
- A *minimal diagnosis* is a set Δ such that Δ is a diagnosis but no $\Delta' \subset \Delta$ is a diagnosis.
- A *conflict set* is a set $C \subseteq \text{COMP}$ such that $\text{SD} \cup \text{OBS} \cup \{\neg ab(c) | c \in C\}$ is inconsistent.

Although in general, SD can be an arbitrary set of first-order sentences, in practice (and also in our approach), SD is assumed to consist of two parts: the *behaviour model* which describes the normal behaviour of each individual component, and the *structure model* which describes the connections between the individual components.

A slight complication arises because the approximate deduction relation BCP_k that we intend to use is only defined for propositional theories. Fortunately, typical behaviour and structure models are in fact function-free and range over finite alphabets. Such first-order theories can be rewritten as equivalent propositional theories using standard methods.

3 DEFINING $\vdash_k^{BCP}$

This section repeats some of the definitions and results from [Dalal, 1996] and [Dalal & Yang, 1997]. They define the family of approximate deduction relations BCP_k.

Boolean Constraint Propagation [McAllester, 1990] is a limited form of deduction which is a variant of unit resolution [Chang & Lee, 1973]. It performs limited deduction in linear time, as follows: given a theory T[1], BCP monotonically expands T by adding literals as follows: in each step, if any single clause in T and all

[1] We will assume all theories to be propositional, and in clausal normal form.

the literals in T taken together entail any other literal (or $\bot$), then this literal (or $\bot$) is added to the theory T. [Dalal, 1996] defines BCP algebraically:

Definition 2 ($\vdash_{BCP}$) *A formula ϕ is inferable from a theory T using BCP, denoted by $T \vdash_{BCP} \phi$, iff $T \cup \{\neg\phi\} =_{BCP} \bot$ via the following rewrite rules for $=_{BCP}$:*

1. $\{\bot\} \cup T \quad =_{BCP} \quad \{\bot\}$
2. $\{(\alpha), (\neg\alpha \vee \alpha_1 \vee ... \vee \alpha_n)\} \cup T \quad =_{BCP} \quad \{(\alpha), (\alpha_1 \vee ... \vee \alpha_n)\} \cup T$

where α and the α_i's are any literals.

BCP is a sound but incomplete (ie approximate) deduction relation:

Example 1

$$\{(P \vee Q), (P \vee \neg Q), (\neg P \vee Q), (\neg P \vee \neg Q)\} \vdash \bot$$
$$\{(P \vee Q), (P \vee \neg Q), (\neg P \vee Q), (\neg P \vee \neg Q)\} \nvdash_{BCP} \bot$$

This shows that $\vdash_{BCP}$ cannot chain on the intermediate result P. Although incomplete in general, BCP is complete for Horn theories. Its incompleteness comes from the inability to use previously inferred clauses during the reasoning process (ie "chaining"):

Example 2 *Let $T_0 = \{(P \vee Q), (P \vee \neg Q), (\neg P \vee S \vee T), (\neg P \vee S \vee \neg T)\}$, then*

- *$T_0 \vdash_{BCP} P$ (by adding $\neg P$ to T_0 and deriving $\bot$ using the first two clauses), and*
- *$T_0 \cup \{P\} \vdash_{BCP} S$ (by adding $\neg S$ and using the last two clauses), but*
- *$T_0 \nvdash_{BCP} S$.*

Allowing BCP to chain on arbitrary clauses would make it sound and complete and would therefore render it uninteresting as an approximate deduction method.

This is the motivation for defining BCP_k [Dalal, 1996], where chaining is allowed, but only on formulae of limited length:

Definition 3 ($\vdash_k^{BCP}$) *For any theory T and clauses ϕ and ψ:*

$$\frac{T \vdash_{BCP} \phi}{T \vdash_k^{BCP} \phi}, \quad \frac{T \vdash_k^{BCP} \psi;\ (T, \psi) \vdash_k^{BCP} \phi}{T \vdash_k^{BCP} \phi} \text{ if } |\psi| \leq k$$

Clearly, when k increases, $\vdash_k^{BCP}$ is allowed to chain on ever more formulae, and will become ever more complete. For example, $T_0 \vdash_0^{BCP} P$, $T_0 \nvdash_0^{BCP} S$ and $T_0 \vdash_1^{BCP} S$. $\vdash_1^{BCP}$ is complete for 2-CNF theories.

4 USING BCP_k FOR DIAGNOSIS

This section contains our results of applying BCP_k to diagnostic reasoning. First we will start with the effect of using a sound but incomplete reasoner for Reiter's definition of diagnostic reasoning. After this, we give a number of examples of diagnostic reasoning using BCP_k as illustration. We continue with some intuitions of using BCP_k in diagnosis and finally we present our results of anytime diagnostic reasoning using approximate BCP.

4.1 USE OF A SOUND, BUT INCOMPLETE REASONER

Using a sound but incomplete approximation of the entailment relation (i.e. BCP_k or the 3-S-approximation from [Schaerf & Cadoli, 1995]) in Reiter's definition of diagnosis has two effects. First, using such an approximation results in less diagnostic problems. The assumption that all components are functioning correctly, together with the observed behaviour and the system description could be classically inconsistent but still consistent under the approximate entailment relation (since $\bot$ could be deducible classically, but not by the weaker approximate entailment):

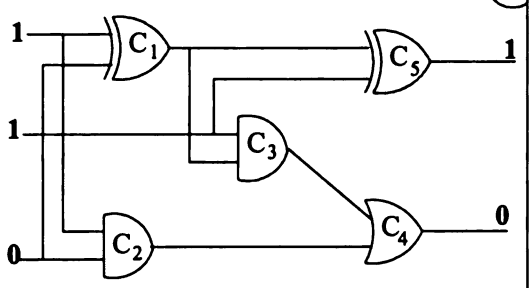
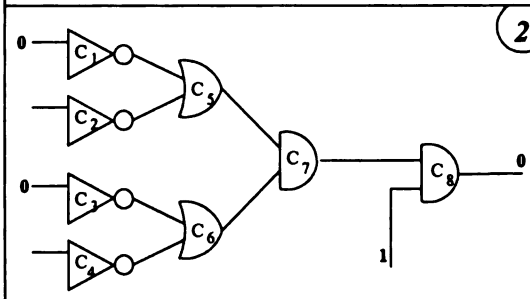
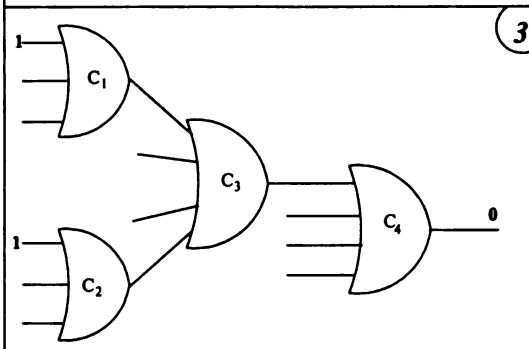
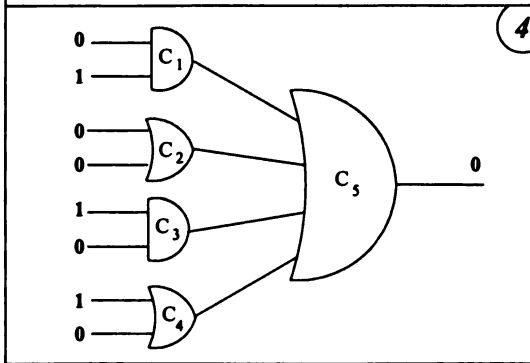
Theorem 1 (Approximate diagnostic problems are classical problems)
For any sound but incomplete entailment relation $\mid\sim$:
if $SD \cup OBS \cup \{\neg ab(c) | c \in COMP\} \mid\sim \bot$
then $SD \cup OBS \cup \{\neg ab(c) | c \in COMP\} \vdash \bot$

The second effect is that diagnoses which are not classical diagnoses are possibly approximate diagnoses. In the classical case such a diagnosis together with the observed behaviour and system description derives $\bot$. Again, because of the incompleteness of the approximation, $\bot$ might not follow in the approximate case and therefore it could be an approximate diagnosis:

Theorem 2 (Every classical diagnosis is an approximate diagnosis)
For any sound but incomplete approximate entailment relation

$$\{\Delta | \Delta \text{ is a classical diagnosis}\} \subseteq \{\Delta | \Delta \text{ is an approximate diagnosis}\}$$

NB: This result only holds if we consider the set of *all* diagnoses. It is no longer true when restricted to the set of minimal diagnoses.

System	value of k	Correct minimal k-diagnoses
1  (From [Reiter, 1987], Figure 1)	$k>=0$	{c1}, {c3,c5}, {c4,c5}
2  (From [Davis, 1988], Figure 20)	$k=0$	$\{\}$
	$k>=1$	$\{c_1\}, \{c_3\}, \{c_5\}, \{c_6\}, \{c_7\}, \{c_8\}$
3 	$k=0,1$	$\{\}$
	$k=2$	$\{c_1\}, \{c_2\}, \{c_3\}, \{c_4\}$
	$k>=3$	$\{c_1,c_2\}, \{c_3\}, \{c_4\}$
4 	$k=0$	$\{c_4\}, \{c_5\}, \{c_1,c_2\}, \{c_1,c_3\}, \{c_2,c_3\}$
	$k=1$	$\{c_4\}, \{c_5\}, \{c_1,c_2,c_3\}$
	$k>=2$	$\{c_4\}, \{c_5\}$

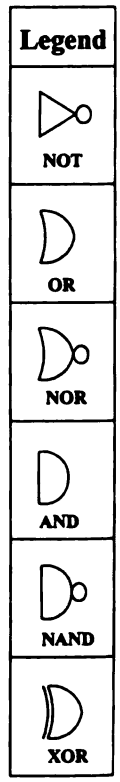


Figure 1: Examples of approximate reasoning in diagnosis using $\vdash_k^{BCP}$

4.2 EXAMPLES AND INTUITION

Figure 1 describes a number of examples of how BCP_k acts in diagnosis. The left-hand side gives the system descriptions and the observations. On the right-hand side the resulting diagnoses are given together with the

k for which they were discovered. Below we describe how the k parameter influences the outcome of the reasoning process.

Combining the definition of diagnosis from [Reiter, 1987] and the definitions of BCP_k allows us to define the notion of a *BCP_k diagnosis*:

Definition 4 (BCP_k diagnosis)
A non-empty candidate diagnosis set $\Delta \subseteq \text{COMP}$ is a *BCP_k diagnosis* iff
$\text{SD} \cup \text{OBS} \cup \{\neg ab(c) | c \in \text{COMP} \setminus \Delta\}$ is BCP_k-consistent.

4.2.1 BCP_0 for a Single Component

The k parameter determines the quality of the computed diagnosis. Its influence is illustrated by means of a simple example. Consider performing diagnosis on a system consisting of only one AND gate in figure 2.

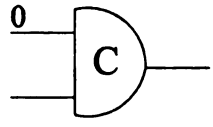


Figure 2: An AND gate with one known input

The system description only contains one behaviour model, and no structural model. The behaviour model of c is[2]:

$$ab\_c \vee \neg in_1\_c\_0 \vee \neg in_2\_c\_0 \vee out\_c\_0$$
$$ab\_c \vee \neg in_1\_c\_0 \vee \neg in_2\_c\_1 \vee out\_c\_0$$
$$ab\_c \vee \neg in_1\_c\_1 \vee \neg in_2\_c\_0 \vee out\_c\_0$$
$$ab\_c \vee \neg in_1\_c\_1 \vee \neg in_2\_c\_1 \vee out\_c\_1$$

The first of these states that given two 0 input values and assuming correct behaviour, the output also equals 0 and similar for the other formulae.

Assume that we are given the observation $\text{OBS} = \{in_1\_c\_0\}$ and that we want to derive that the output of this component is 0: out_c_0, assuming it is working correctly: $\neg ab\_c$.

$$SD \cup \{in_1\_c\_0\} \cup \{\neg ab_c\} \vdash_0^{BCP} out\_c\_0$$

Adding $\neg out\_c\_0$ to the theory, and performing unit resolution on these literals and the behaviour model results, among others, in:

$$\neg in_2\_c\_0$$
$$\neg in_2\_c\_1$$

[2]Notations such as out_c_0 are the propositional translation of the first-order formula $out(c) = 0$.

As a result of the translation into propositional logic, the clause $in_2\_c\_1 \vee in_2\_c\_0$ is in SD. This can be used to derive the empty clause using the previous two literals, establishing the desired conclusion. This shows that $k = 0$ allows some inference over the behaviour of components.

4.2.2 Connections Between Components

To allow computed values to spread throughout an entire system, they have to be propagated from one component to another.

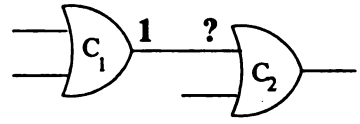


Figure 3: Propagating values from one component to another

Consider the example in figure 3. An output value of 1 is known for component c_1. To propagate this value towards c_2, the value has to go through the link connecting these two components. After propositional translation, the formulas in the structure model that describe the link in the above example look like (For all v in some defined set VALUES):

$$\neg out\_c_1\_v \vee in_1\_c_2\_v \text{ and } \neg in_1\_c_2\_v \vee out\_c_1\_v$$

If for example $out\_c_1\_1$ is known, then also its counterpart, $in_1\_c_2\_1$ is BCP inferable. In other words, if a value assignment to one side of a link can be inferred for a certain k, the value on the other side of the link can be inferred as well.

4.2.3 BCP_0 for Multiple Components

Consider the following example:

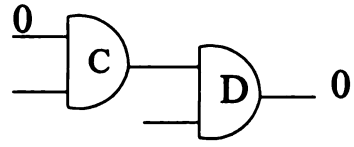


Figure 4: Reasoning about multiple components

As before, we try to use BCP_0 to derive that the output of this circuit is 0:

$$SD \cup \{in_1\_c\_0\} \cup \{\neg ab\_c, \neg ab\_d\} \vdash_0^{BCP} out\_d\_0$$

The SD for this circuit consists of clauses for both and-gates, augmented with the clause stating the connection between the two. Unlike the single-component

case from section 4.2.1, no new literals can be inferred. The only possible steps yield the following binary clauses (resulting from the description of C):

$$\neg in_2\_c\_0 \vee out\_c\_0$$
$$\neg in_2\_c\_1 \vee out\_c\_0$$

This states that regardless of its second input value, C's output will be 0. Although this is classically derivable, such a combination of two binary clauses is not allowd under BCP_0. As a result, BCP_0 is not able to show that the output of the entire circuit is 0 either.

Observation: Notice that BCP_0 would not have failed on this example if either of the other two input values had been given. Classically, these values are irrelevant to the output of the circuit. For BCP_0 however, this redundant information would have been just enough to achieve the desired conclusion. For example, knowing either $in_2\_c\_0$ or $in_2\_c\_1$ would allow further reduction of one of the above two binary clauses, leading to out_c_0.

4.2.4 BCP_1 for Multiple Components

Now we will show that unlike BCP_0, BCP_1 is able to derive the output of the above example:

$$SD \cup \{in_1\_c\_0\} \cup \{\neg ab\_c, \neg ab\_d\} \vdash_1^{BCP} out\_d\_0 \quad (1)$$

According to definition 3, we are allowed to hypothesise a formula of length k (here $k = 1$, ie a literal), to prove this hypothesis, and then exploit this result in the subsequent reasoning. Such chaining on formulae of length 1 is exactly what was disallowed under BCP_0. In the example of fig. 4 the obvious literal to use is out_c_0. This is easily established under BCP_1 (in fact, we saw it already followed under BCP_0). We are then allowed to add out_c_0 to the left hand side of (1). This enables a similar inference on component D to derive the required result.

4.2.5 BCP_k for more complex components

Notice that the circuit from fig. 4 is equivalent to a single AND-component with three input gates: The

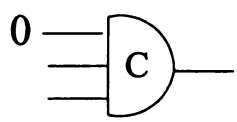


Figure 5: Reasoning about multiple components

same results as above hold for this complex component: BCP_0 cannot derive out_c_0, while BCP_1 can. BCP_0 cannot further reduce the following set of binary clauses:

$$\neg in_2\_c\_1 \vee \neg in_3\_c\_1$$
$$\neg in_2\_c\_1 \vee \neg in_3\_c\_0$$
$$\neg in_2\_c\_0 \vee \neg in_3\_c\_1$$
$$\neg in_2\_c\_0 \vee \neg in_3\_c\_0$$

As before, classically this set of clauses is inconsistent (as desired), but BCP_0 cannot derive this. BCP_1 is able to derive this inconsistency by first hypothesising (for instance) $in_2\_c\_0$.[3] This hypothesis can be proved using the clause $in_2\_c\_0 \vee in_2\_c\_1$ (resulting from the translation to propositional logic) and the same clause for in_3_c, in combination with the first two binary clauses above. Now we can chain on this result ($in_2\_c\_0$) which gives the required empty clause by combining the last two binary clauses above with the exclusion clause for in_3_c.

Summarising: although BCP_0 can be used to reason about single simple components, and about the connections between them, higher values of k are required to reason about more complex circuits. These intuitions will be made more precise in the subsequent theorems.

5 THEOREMS ON DIAGNOSTIC REASONING WITH BCP_k

After presenting the intuition of using BCP_k, this section presents the formal results about the effects of using BCP_k in diagnostic reasoning. After some initial and easy results, we will be mostly concerned with upperbounds on values for k: for which values of k do classical diagnosis and approximate diagnosis coincide. Some of these results are quite surprising and yield much lower values for k than might be expected.

5.1 EFFECTS OF THE INCOMPLETENESS OF BCP_k

In section 4.1, we mentioned the general effect of a sound and incomplete approximation of the entailment relation for consistency based diagnosis. As a special case, the following holds for BCP_k:

Theorem 3 (Every k diagnostic problem is also a $k+1$ diagnostic problem)
if $SD \cup OBS \cup \{\neg ab(c) | c \in COMP\}$ is BCP_k-inconsistent then $SD \cup OBS \cup \{\neg ab(c) | c \in COMP\}$ is BCP_{k+1}-inconsistent

[3] In fact, any of the literals in these clauses would have done the trick.

Theorem 4 (Increasing k leads to fewer diagnoses)
For any SD, COMP and OBS:

$$\{\Delta|\Delta \textit{ is a } BCP_{k+1} \textit{ diagnosis}\} \subseteq \{\Delta|\Delta \textit{ is a } BCP_k \textit{ diagnosis}\}$$

As already stated after theorem 2, this is only true for the set of all diagnoses. Examples 3 and 4 from figure 1 show that this result does not hold when restricted to minimal diagnoses. For example, in example 3 from the figure, $\{c_1, c_3\}$ is a diagnosis at $k = 2$ as well, albeit not a minimal not.

5.2 AN UPPERBOUND FOR k ON SINGLE COMPONENTS

In general, the values that BCP_k can derive depend on the value of k (obviously), the complexity of the component (section 4.2.5) and the set of known input or output values of the component (the final observation from section 4.2.1). The following lemma provides a precise characterization of the relation between the value of k and the formulas that can be inferred about a single component.

Lemma 1 (Upperbound for k for a single component)
Consider a system $(SD, COMP)$, a set VALUES and an observation set OBS. For every component $c \in COMP$ having n links going in or out, of which $m(\leq n)$ values are elements of OBS and a literal ϕ, which is either $\bot$ or an assignment of a value $v \in VALUES$ to an input or output of c, the following holds:

If $k \geq n - m - 1$ and $SD \cup OBS \cup \{\neg ab\_c\} \vdash \phi$, then $SD \cup OBS \cup \{\neg ab\_c\} \vdash_k^{BCP} \phi$

Next, we will use this lemma to establish an upperbound for the value of k that is required to reason about entire circuits instead of only single components.

5.3 AN OBSERVATION-INDEPENDENT UPPERBOUND for k

It is possible to derive an upper bound of k for a given system (SD,COMP) that is needed to compute classical diagnoses for an arbitrary diagnostic problem. A surprising result is that this upperbound depends only on the types of the components in COMP, and not and not on how they are connected. This upperbound is computed as follows:

First we apply lemma 1 to each component with $m = 0$ (no observations, ie. the worst case), giving as value $n - 1$ for each component, where n is the number of links going in or out. We then take the maximum of all these values:

Definition 5 $k_{max}(SD, COMP)$ = *maximum of* $n - 1$ *over all components in COMP*

Theorem 5 (Invariance of k with system size)
Let SD and SD' be two system descriptions which contain the same types of components (ie they only differ in the number of components and how these components are connected, their structure model), then

$$k_{max}(SD) = k_{max}(SD')$$

Proof sketch: Lemma 1 ensures that a component can be correctly simulated at $k = n - 1$ (taking $m = 0$, the worse case). An entire circuit is simply the union of the component descriptions (we can ignore the connection clauses, since section 4.2.2 showed that the connection clauses can be dealt with even by $k = 0$). The k required for an entire circuit is therefore not higher than the k for the most complex component, ie. the maximum of $n - 1$ over all components. □

Lemma 1 ensures that for this value of k we will obtain only the classical diagnoses.

5.4 AN OBSERVATION-DEPENDENT UPPERBOUND for k

Although the previous result stated that the maximally needed value of k is independent of the system interconnection, this value can in general be lowered when information about the given observations is exploited. We will now derive the maximal value for k that is needed for a given SD and a set of observations OBS. For this value of k or higher, BCP_k diagnoses coincide with the classical results.

The characterisation of the upper bound of k is based on conflict sets.

Definition 6
- *Given SD, OBS and a minimal conflict set mcs, ie.* $SD \cup OBS \cup \{\neg ab(c)|c \in mcs\} \vdash \bot$, *then* $k_{needed}(mcs, SD, OBS)$ *is defined as the minimal k for which* $SD \cup OBS \cup \{\neg ab(c)|c \in mcs\} \vdash_k^{BCP} \bot$;
- *Given all minimal conflict sets $mcs_1 \ldots mcs_n$ for a diagnostic problem (SD,COMP,OBS), k^* is defined as follows:*

$$k^* = max_{1 \leq i \leq n}(k_{needed}(mcs_i, SD, OBS))$$

This definition of k^* allows the following theorem:

Theorem 6 (When BCP_k diagnoses equal classical diagnoses)

Δ is a BCP_{k^*} diagnosis for $(SD, COMP, OBS)$ iff *Δ is a classical diagnosis for $(SD, COMP, OBS)$.*

Theorem 6 is of great importance since it provides an upper bound for k. No k greater than k^* needs to be tried, since the results can not get any better.

The question remains how the minimal k can be discovered for which $\text{SD} \cup \text{OBS} \cup \{\neg ab(c) | c \in mcs\} \vdash_k^{BCP} \bot$, which actually comes down to checking how the conflicting observations can be propagated through the part of the system that is the minimal conflict set. The behaviour of components that are not in *mcs* does not matter: they are not assumed to be working correctly and no conclusions can be drawn for them. For this, first the minimal conflict sets have to be found. The task of determining the minimal conflicts sets is of the same complexity as performing classical diagnostic reasoning; from the minimal conflict sets, the classically correct diagnoses can very easily be determined.

As a result, the upper bound k^* from the above theorem is not one that can be effectively exploited: even though we know it exists, determining its actual value is of the same complexity as performing (classical) diagnosis in the first place.

Of course, one expects that the observation-dependent upperbound k^* is sharper than k_{max}, the observation-independent upperbound. This is indeed the case:

Theorem 7 *For any problem (SD,COMP,OBS):*

$$k_{max}(SD) \geq k^*$$

The following table shows the values of k_{max} and k^* for all the examples from figure 1:

Example no.	k_{max}	k^*
Example 1	2	0
Example 2	2	1
Example 3	3	3
Example 4	3	2

Theorem 7 states that k_{max} is only a sufficient but not necessary upperbound. On the other hand, k_{max} is easily computable (linear in the size of the circuit) while the cost of computing k^* is of the same order as performing classical diagnosis in the first place. The figures from the above table show that the cheap upperbound k_{max} does not overshoot the optimal value by very much, at least on the small examples from figure 1.

5.5 BINARY CIRCUITS CAN BE DIAGNOSED WITH k=1

Systems consisting of the traditional one- or two-input gates such as AND, OR and NOT gates play a central role in most literature on formalising diagnosis. Theorem 8 states a surprising result about such systems consisting only of components with at most two inputs.

Theorem 8 ($k = 1$ suffices for diagnosing binary circuits)
For all electronic circuits consisting of components with no more than two inputs, Δ is a BCP_1 diagnosis iff *Δ is a classical diagnosis.*

Proof sketch: Even full classical entailment can only derive results about components with at least one known input or output signal. Choose such a component. Lemma 1 guarantees that for this component BCP_1 can derive all classically valid results (since $n = 3, m \geq 1$). Therefore, for this component, BCP_1 reasoning completely emulates classical reasoning. This argument can be repeated for every subsequent classical reasoning step. □

This is a quite surprising result, because apparently all diagnostic problems for this kind of systems can be solved with an entailment relation as simple as BCP_1, regardless of the size or structural complexity of the system.

5.6 RECOGNISING DIAGNOSTIC PROBLEMS WITH k=0

As can be seen from examples 2 and 3 from fig. 1, sometimes BCP_k is not able to even recognise the existence of a diagnostic problem. It is not able to derive inconsistency, and suggests the empty diagnosis (ie. no component is functioning incorrectly. The following theorem says that in certain cases, BCP_k is guaranteed to recognise a diagnostic problem if there is one.

Theorem 9 (recognising diagnostic problems at k=0)
For all electronic circuits for which all the inputs are known, Δ is a BCP_0 diagnosis iff *Δ is a classical diagnosis.*

Proof sketch: For recognising a diagnostic problem, we must prove

$$SD \cup \{\neg ab\_c | c \in COMP\} \cup \text{OBS} \vdash_0^{BCP} \bot \quad (2)$$

For some components c, all inputs will be known (since all inputs to the circuit are assumed to be known). The description of components consists of clauses of

the following form:

$$ab\_c \vee \neg in_1\_c\_v \vee ... \vee \neg \in_n \_c\_v \vee out\_c_v$$

Fro the literals given on the left-hand side of (2), this can be reduced to the literal *out_c_v* (because all inputs of c are known). This amounts to "removing component c from the circuit". This process can be repeated until no components are left. At that time, only literals are left, and inconsistency can be derived by BCP_0. □

The strength of the theorem is that it applies to systems of arbitrary complexity. The weakness of this theorem is that it only applies when all input signals are known. If n inputs values are unknown, then the theorem can be applied for all of the 2^n possible open combinations. If n is not too large, 2^n applications of unit resolution may still be more efficient then a single application of a general resolution procedure.

6 CONCLUSIONS

In this paper we have shown how approximate boolean constraint propagation can be used for approximate diagnostic reasoning.

We have taken the well-known definition of Reiter's consistency based diagnosis, and replaced the logical entailment in its definition with a sound but incomplete approximation of the logical entailment. Using such an approximated version less diagnostic problems are recognized, but when a diagnostic problem is recognized more diagnoses are found. We were able to characterise the cheapest versions of approximate BCP which allows single components and entire circuits to be diagnosed correctly. From these upperbounds surprisingly low values follow which are needed to correctly diagnose many of the typical examples from the literature. A particularly interesting property that we discovered is that the complexity level at which approximate diagnoses coincide with classical diagnoses is entirely determined by the nature of the individual components, and not by either the size or the structural complexity of the overall device.

The question remains whether k is a good indicator for the complexity of diagnostic problems. For traditional diagnosis of binary circuits, k is definitely not a good indicator: theorem 8 states that $k = 1$ already captures all complexity. This leaves $k = 0$ as the only approximating step. For more complex components with more input- and output gates, the value where k-diagnosis becomes classical is higher. In principle, makes it possible to approximate more gradually the classical case with increasing k. This would make k an attractive indicator of diagnostic complexity in the case of complex components. Whether or not this is the case cannot be established by the small examples that we have studied so far.

Related work In [tenTeije & vanHarmelen, 1996], we already studied approximate diagnosing by using the approximate entailment from Schaerf and Cadoli [Schaerf & Cadoli, 1995] in several definitions of diagnosis. Compared with this earlier work, one of the drawbacks of using BCP_k for approximation is that we are not able to give guidelines for a good choice for k. We can only say that one should start at $k = 0$ and must increase k until the upperbound has been reached. For the Cadoli/Schaerf approximation on the other hand, we were able to give strategies for good choices for the parameter that determines cost and quality of the approximation [tenTeije & vanHarmelen, 1997].

A second point in favour of the Schaerf/Cadoli approximations is the desired gradual behaviour of an approximate diagnostic method. For the Schaerf/Cadoli approximation, we have observed such gradual behaviour (see the examples in [tenTeije & vanHarmelen, 1996]), while for BCP_k this question is still open, and the initial evidence is discouraging. A further problem with BCP_k is that even the initial step in the approximating process ($k = 0$) is already quite strong (BCP_0 is complete for Horn theories), thereby removing much scope for approximating steps which were possible under the Schaerf/Cadoli approximation.

Other work in approximate diagnosis is done by Pos [Pos, 1993]. She takes a more algorithmic approach and applies the approach proposed in [Russel & Zilberstein, 1991] to diagnostic reasoning. The diagnosis task is decomposed in a number of subtasks (conflict generation, discrimination and conversion) and for each of these a number of anytime algorithms are proposed which should be chosen and scheduled. She discusses three methods for scheduling several anytime algorithms and applies these to diagnosis. This work is complementary to our approach, which deals with declarative specifications only.

Future work As mentioned above, it would be an attractive anytime behaviour if with every increase of k the quality of the diagnoses would increase a little bit. However, in our rather small experiments (both in size and in number), we often observe a behaviour that is the opposite of this: for small values of k the approximate diagnoses remained the same, only to suddenly jump to the entire set of classical diagnoses. The next step in our study of BCP_k is to do more experiments for

testing under which conditions performing BCP_k diagnosis would result in better anytime behaviour. Another way of extending our experiments is to use more complicated behaviour models. The same should be done for investigating the tightness of the upperbound k_{max} on more realistic examples. In our examples we considered only models of the correct behaviour of components and therefore we are allowed to use the minimal consistency based diagnosis hypothesis [McIlraith, 1994]. A new subject of study is whether our results are depend on this minimal consistency based diagnosis hypothesis.

In [tenTeije & vanHarmelen, 1996], we showed that the use of a unsound but complete approximation gives useful results for performing diagnosis. Therefore another direction of further research is to look for an unsound but complete BCP approximation.

Acknowledgements

We are grateful to Wouter Teepe for providing an implementation of BCP_k, and to Rineke Verbrugge for supervising him during this work and for useful discussions in the early stages of this work.

References

[Boddy & Dean, 1989] M. Boddy and T. Dean. Solving time-dependent planning problems. In IJCAI–89.

[Bylander *et al.*, 1991] T. Bylander, D. Allemang, M. Tanner, and J. Josephson. The computational complexity of abduction. *Artificial Intelligence*, 49:25–60, 1991.

[Chang & Lee, 1973] C. Chang and R.C. Lee. *Symbolic Logic and Mechanical Theorem Provering.* Academic Press, London, 1973.

[Dalal & Yang, 1997] M. Dalal and L. Yang. Preliminary emperical results on anytime propositional reasoning (abstract). In *AAAI97 workshop*, 1997.

[Dalal, 1996] M. Dalal. Semantics of anytime family of reasoners. In ECAI'96, pages 360–364, 1996.

[Davis & Hamscher, 1988] R. Davis and W. C. Hamscher. Model-based reasoning: Troubleshooting. In H. E. Shrobe, editor, *Exploring Artificial Intelligence*, pages 297–346. Morgan Kaufmann, 1988.

[Dean & Boddy, 1988] T. Dean and M. Boddy. An analysis of time-dependent planning. In AAAI'88, pages 49–54, 1988.

[Green, 1969] C. Green. Application of theorem proving to problem solving. InIJCAI'69, pages 219–239, 1969.

[McAllester, 1990] D. McAllester. Truth maintenance. In AAAI'90, pages 1109–1116, 1990.

[McIlraith, 1994] S. McIlraith. Further contributuins to characterizing diagnosis. *Annals of Mathematics and AI*, 11(1–4), 1994. Special issues on Model-based diagnosis.

[Pos, 1993] A. Pos. Time-constrained model-based diagnosis. Technical report, University of Twente, department of computer science, 1993. Master's thesis.

[Reiter, 1987] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–96, 1987.

[Rosenschein, 1981] S. Rosenschein. Plan synthesis: a logical perspective. In IJCAI'81, pages 331–337, 1981.

[Russel & Zilberstein, 1991] S. Russel and S. Zilberstein. Composing real-time systems. In IJCAI'91, pages 212–217, 1991.

[Schaerf & Cadoli, 1995] M. Schaerf and M. Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74(2):249–310, April 1995.

[tenTeije & vanHarmelen, 1996] A. ten Teije and F. van Harmelen. Computing approximate diagnoses by using approximate entailment. In KR'96, 1996. Morgan Kaufman.

[tenTeije & vanHarmelen, 1997] A. ten Teije and F. van Harmelen. Exploiting domain knowledge for approximate diagnosis. In IJCAI'97, pages 454–459, August 1997.

[vanHarmelen & tenTeije, 1995] F. van Harmelen and A. ten Teije. Approximations in diagnosis: motivations and techniques. In A. Levy and P. Nayak, editors, *Proceedings of SARA-95, Symposium on Abstraction, Reformulation, and Approximation*, pages 149–155, Aug 1995.

[Zilberstein & Russell, 1996] S. Zilberstein and S.J. Russell. Optimal composition of real-time systems. *Artificial Intelligence*, 82(1–2):181–213, 1996.

Generation of Diagnostic Knowledge by Discrete-Event Model Compilation

Gianfranco Lamperti **Marina Zanella**
Dipartimento di Elettronica per l'Automazione
Università degli Studi di Brescia
Via Branze 38, 25123 Brescia, Italy
{lamperti,zanella}@ing.unibs.it

Abstract

It is commonplace considering model-based and rule-based diagnosis as tasks with different ontological status, where the former is grounded on design knowledge and 'first principles', while the latter is merely based on empirical associations. In contrast with this view, we present a method that bridges model-based and rule-based diagnosis of a class of distributed discrete-event systems, which are networks of communicating automata that integrate both synchronous and asynchronous behavior. Recent research on the subject has mainly concentrated on the on-line diagnosis of such systems. This paper, instead, focuses on off-line activities, by showing how system models can be automatically compiled into a set of diagnostic rules, these being associations between matching conditions for observable events and diagnoses. Given a system and its observation, namely a diagnostic problem, diagnosis is generated at runtime by filtering out those rules matching the system observation. This way, both the accuracy of the model-based approach and the simplicity and efficiency of the reasoning mechanism of the rule-based counterpart are saved. Furthermore, a hybrid technique can be conceived, that integrates on-line and off-line diagnosis within a single procedure, which is scalable, hierarchical, and amenable to parallel and distributed computation.

1 INTRODUCTION

Over the last few years, considerable research effort has been spent on diagnosis of discrete event systems (DESs), such as digital networks, communication networks, and industrial protection apparatus [Cordier and Thiébaux, 1994, Sampath et al., 1995, Sampath et al., 1996, Laborie and Krivine, 1997, Rozé, 1997, Lamperti and Pogliano, 1997, Baroni et al., 1997, Baroni et al., 1998]. A new approach has been recently introduced [Baroni et al., 1999, Lamperti and Zanella, 1999, Lamperti et al., 1999, Lamperti et al., 2000], which features compositional modeling and does not require the (either manual or automatic) creation of any global diagnoser.

A distributed DES is modeled as a network of automata [Brand and Zafiropulo, 1983], each automaton describing the complete behavior of a component and communicating with other automata by means of links, where each link is either an asynchronous buffer or a synchronous channel.

An event occuring in the outside world triggers a *reaction* of the system, that is, a sequence of state transitions, during which the outside world may generate further events directed to the system, and system components may generate events directed to the external world and/or to other components. When the reaction finishes, the system becomes *quiescent*, in other words, it does not change its state until a new event occurs in the external world. State transitions generally produce some observable events, called *messages*.

The diagnostic method proposed in the previous works encompasses *on-line* activities only, which are carried out once the reaction of the system has finished. Such method involves the reconstruction of the system reaction on the basis of the observation gathered during the reaction. The *observation* of the system is a set of totally temporally ordered sequences of messages, called *observation items*, each pertaining to a group of components belonging to a given partition of the set of all system components. Several (possibly an unbound

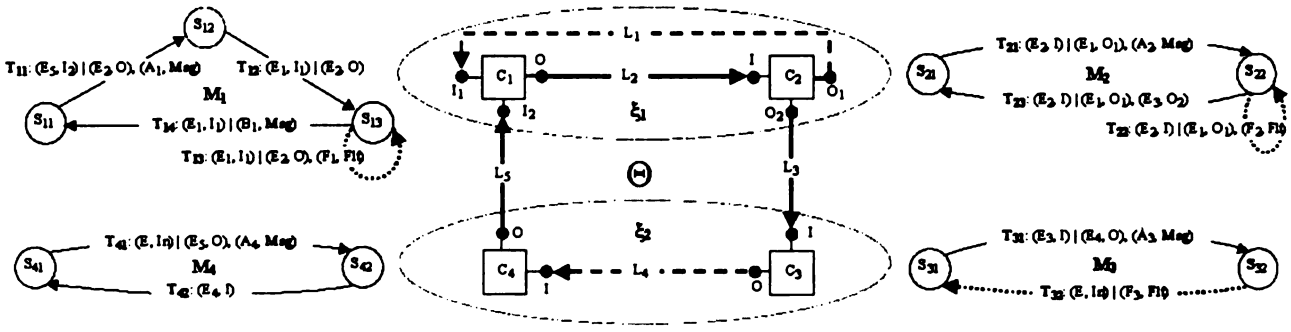


Figure 1: System Θ and the Behavioral Models of its Components

number of) alternative histories of the system can be obtained and represented by means of a finite graph, the *active space*, where each *history* is a sequence of state transitions that originates from the known initial state, yields all the messages of the observation, according to their temporal ordering, and ends in a quiescent state. Each transition in a reconstructed history is either *nominal* or *faulty*, the latter pointing out a specific fault in the relevant component. Hence a (possibly empty) set of faulty components, namely a *diagnosis*, is associated with each reconstructed history and several alternative diagnoses can possibly be derived from distinct histories.

Notice that, in spite of the fact that the number of histories is potentially unbound, the set of different diagnoses is always finite, being a diagnosis an element of the powerset of the components of the system.

In this paper an approach is proposed, which, although based on the same modeling primitives and reasoning mechanisms as the previous one, encompasses also *off-line* activities, so as to speed up the diagnostic process.

Off-line activities largely consist in preprocessing the models of system components in order to automatically generate a set of *diagnostic rules*. Specifically, after the system has been decomposed into (not necessarily disjoint) sub-systems, or *clusters*, a set of diagnostic rules can be generated, possibly in parallel, for each cluster. Once the reaction has finished, the candidate diagnoses of a cluster whose set of diagnostic rules is available are determined on-line by checking each and every diagnostic rule against the actual cluster observation.

In the remainder of the paper, Section 2 informally introduces the diagnostic approach, which is then formalized in Section 3. Algorithms are presented in Section 4. A discussion and a comparison with the state of the art is outlined in Section 5. Section 6 concludes the paper.

2 APPROACH

Based on a very simple example, Sections 2.1 and 2.2 summarize the modeling foundations of the approach, which are formally described in [Baroni et al., 1999, Lamperti and Zanella, 1999, Lamperti et al., 2000]. Section 2.3, instead, briefly introduces the off-line method which is the original contribution of this paper.

2.1 SYSTEM

DESs are composite systems whose *components* are endowed with input and output *terminals*, which are connected with each other through directed connection *links*. On the center of Figure 1, a system Θ is represented, that embodies four components, namely, $C_1, \ldots, C_4$, and five links, $L_1, \ldots, L_5$. The initials of the identifiers of input and output terminals are I and O, respectively.

During the reaction, components generate events, each of which is either directed to the external world, to a neighboring component, or to a passive external observer. In addition, the external world may asynchronously generate events directed to the system. Every event directed from a component to another one is transmitted on a link, which is either *synchronous* or *asynchronous*. In the former case, the event is available to the neighboring component as soon as it is transmitted and is immediately consumed. In the latter case, the event is temporally saved in the link. Each link is characterized by a *capacity*, which is the (finite) maximum number of events that can be buffered within the link, and by a management policy (called *saturation mode*). In the example, it is assumed that each one of the links displayed in solid lines, namely L_2, L_3 and L_5, is asynchronous, has a capacity equal to one, and a policy such that a new event can be generated only if there is room for it within the link. Instead, links L_1 and L_4, depicted in dashed lines, are assumed

to be synchronous. The *dangling set* of a link L_i is the actual set of events buffered in L_i at a given time.

A model describes the complete behavior of a component type. On the left and on the right of Figure 1, each M_i is the model of C_i. Incidentally, system Θ includes one instance of each component type: in general, a system may include several instances of any component type.

A model is a finite automaton, expressing output events as a function of the input event and the internal state. For example, model M_1 includes three states, namely S_{11}, S_{12}, and S_{13}, and four transitions, $T_{11},\ldots,T_{14}$. Transition T_{11} is $S_{11} \xrightarrow{\alpha|\beta} S_{12}$, where $\alpha = (E_5, I_2)$ and $\beta = \{(E_2,O), (A_1, Msg)\}$. This means that T_{11} is triggered by the input E_5 on terminal I_2, and generates the set of output events in β. In particular, output event (E_2, O) implies that E_2 is buffered within the link connected with terminal O. An output sent to the *message terminal*, Msg, such as (A_1, Msg), is a *message*. Only outputs on terminal Msg are observable. Transition T_{11} is *observable*, insofar as it involves a message, while T_{12} is *silent*. Transition T_{13} is *faulty* since it generates an output F_1 on Flt, the *fault terminal*. In Figure 1, faulty transitions are represented by dotted arrows. Outputs sent to Flt are a modeling artifice to specify faulty behaviors. A transition can generate one output at most on each output terminal. Each component is implicitly endowed with a *standard input* (virtual) terminal, In, which is meant to connect the component with the external world.

2.2 REACTION

A system, while operating, is either *quiescent* or *reacting*. A quiescent system has no buffered events within its links and it neither changes its state nor produces any output event. A quiescent system becomes reacting upon the arrival of an event on the standard input of a component.

Before the reaction starts, every component is waiting in a state; afterwards, during the reaction, it undergoes a sequence of transitions, namely a *history* of the component, which connects the initial state with a final state. For example, assuming that the initial state of C_1 is S_{11}, a possible history of C_1 is $\langle T_{11}, T_{12}, T_{13}, T_{13}, T_{14}, T_{11}\rangle$. The sequence of messages generated by a component according to a given history is the *observation* of the component. In the above example, the history of C_1 produces the observation $obs(C_1) = \langle A_1, B_1, A_1\rangle$.

A *cluster* is a connected (sub-)graph incorporating one or several system components and all the links among them. A system is a cluster itself. The concepts of history and observation can be straightforwardly extended to clusters. The history of a cluster is a sequence of transitions, each one relevant to a component included in the cluster, which connects the initial state of the cluster (that is, the composition of the initial states of its components) with a final state. A possible history of system Θ is $h(\Theta) = \langle T_{41}, T_{11}, T_{21}, T_{12}, T_{22}, T_{13}, T_{22}, T_{13}, T_{23}, T_{14}, T_{31}, T_{42}, T_{32}\rangle$, where the initial states for C_1, C_2, C_3, and C_4 are S_{11}, S_{21}, S_{31}, and S_{41}, respectively. The final state of the cluster history, which is the state of the cluster at the end of the reaction, is a quiescent state, that is, a state wherein the dangling sets of all the links included in the cluster are empty.

The messages generated according to the cluster history represent the cluster observation. In particular, an *ordering partition* is associated with each cluster, which is a partition of the set of all cluster components. Each part includes cluster components whose messages are received by the external observer in total temporal order. No ordering, instead, is assumed among the messages of distinct groups of components. Therefore, the observation of a cluster is a composition of totally temporally ordered sequences of messages, called *observation items*, each pertaining to a part of the ordering partition of the cluster. For instance, assume that the ordering partition of system Θ is $\mathcal{P}(\Theta) = \{\mathcal{C}, \mathcal{C}'\}$, where $\mathcal{C} = \{C_1\}$ and $\mathcal{C}' = \{C_2, C_3, C_4\}$. Then, the observation of the system is given in terms of two observation items. For instance, the observation entailed by $h(\Theta)$ is $OBS(\Theta) = (obs(\mathcal{C}), obs(\mathcal{C}'))$, where $obs(\mathcal{C}) = \langle A_1, B_1\rangle$ and $obs(\mathcal{C}') = \langle A_4, A_2, A_3\rangle$.

2.3 OFF-LINE METHOD

As detailed in [Lamperti and Zanella, 1999], the central task of the diagnostic technique, namely *behavior reconstruction*, is in charge of generating an *active space*, that is, a graph-based representation of the (possibly boundless) set of the histories of a system that entail a given observation. This task is performed by following a *reconstruction plan*, which is a directed acyclic graph generated by the *reconstruction planning* phase.

The goal of the off-line approach is to considerably reduce the amount of processing needed on-line by the diagnostic process.

According to the off-line method, the phases of reconstruction planning and behavior reconstruction are carried out off-line, and are followed by a compilation of the obtained active spaces. Since the actual observa-

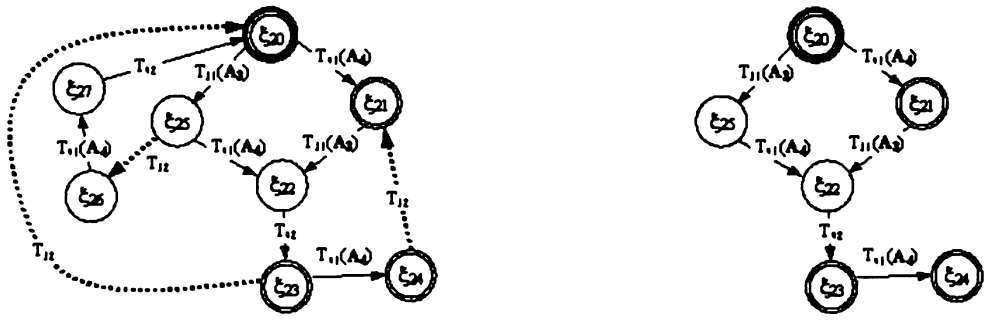


Figure 2: Universal Space of Cluster ξ_2 (left) and a Relevant Subspace (right)

tion is not available off-line, only active spaces whose associated observations are unavailable, namely *universal spaces*, can be generated. Given an initial state ξ_0 of a cluster ξ and assuming that the observation of ξ is unavailable, the consistent histories are all the histories rooted in ξ_0 which are compliant with any possible observation of ξ. The universal space of ξ rooted in ξ_0 is denoted by $\mathcal{U}(\xi, \xi_0)$. Universal spaces may be relevant to any cluster included in the system and possibly to the whole system. Realistically, the selection of the clusters whose universal spaces have to be generated off-line may be driven by domain-dependent criteria. In fact, the selected clusters should be those on which attention is most frequently focused for diagnostic purposes. In addition, since the unavailability of the observations is bound to widen the search spaces, clusters should be selected based also on the practical feasibility of the reconstruction of their universal spaces.

With reference to Figure 1, assume that a selected cluster for off-line processing is ξ_2, whose ordering partition is the singleton $\{\{C_3, C_4\}\}$. The universal space $\mathcal{U}(\xi_2, \xi_{20})$, where $\xi_{20} = (S_{31}, S_{41})$, is shown on the left of Figure 2, where final states are denoted by double circles, while state ξ_{20}, which is both initial and final, has two circles, one of which in bold. The identifier of each observable transition is associated with the message it generates (e.g., $T_{31}(A_3)$). Once the universal space relevant to a cluster has been generated, all the possible diagnoses have to be extracted. In general, only a subset of all the possible combinations of components corresponds to the diagnoses entailed by $\mathcal{U}(\xi, \xi_0)$. These are denoted by $\Delta(\xi, \xi_0)$. In particular, two diagnoses can be identified within $\mathcal{U}(\xi_2, \xi_{20})$, namely $\Delta(\xi_2, \xi_{20}) = \{\delta_1, \delta_2\}$, where $\delta_1 = \emptyset$ and $\delta_2 = \{C_3\}$. The empty diagnosis δ_1 is entailed by the six histories represented in the subgraph displayed on the right of Figure 2 and listed in Table 1.

The subgraph displayed on the right of Figure 2 is called a *route* of $\mathcal{U}(\xi, \xi_0)$ and is associated with a single diagnosis. In general, if $\delta \in \Delta(\xi, \xi_0)$, route $\rho(\delta)$ is the minimal subspace of $\mathcal{U}(\xi, \xi_0)$ such that ρ is rooted in ξ_0 and each history in $\mathcal{U}(\xi, \xi_0)$ entailing δ is also a history in ρ. Thus, $\mathcal{U}(\xi, \xi_0)$ includes as many routes as the number of diagnoses relevant to $\mathcal{U}(\xi, \xi_0)$, that is, if $\Delta(\xi, \xi_0) = \{\delta_1, \ldots, \delta_n\}$, $\mathcal{U}(\xi, \xi_0)$ can be viewed as the union of n routes, one for each diagnosis δ_i. Each route is the graph representation of a (possibly infinite) set of histories, denoted by $\mathcal{H}^+(\rho(\delta))$. However, generally speaking, not all of the histories in $\mathcal{H}^+(\rho(\delta))$ entail the diagnosis δ. In particular, there exists a proper (possibly empty) subset $\mathcal{H}^-(\rho(\delta))$ of $\mathcal{H}^+(\rho(\delta))$ such that each history in $\mathcal{H}^-(\rho(\delta))$ (called a *spurious* history) entails a diagnosis $\delta' \subset \delta$. We denote with $\mathcal{H}(\rho(\delta))$ the difference $\mathcal{H}^+(\rho(\delta)) - \mathcal{H}^-(\rho(\delta))$, in other words, $\mathcal{H}(\rho(\delta))$ incorporates all and only the histories in $\rho(\delta)$ which entail δ.

In our example, $\mathcal{H}^-(\rho(\delta_1)) = \emptyset$. The existence of spurious histories is evident by looking at the route relevant to diagnosis $\delta_2 = \{C_3\}$, which corresponds to the whole universal space displayed in Figure 2. In fact, all the six histories in $\mathcal{H}(\rho(\delta_1))$ are as well included in $\mathcal{H}^+(\rho(\delta_2))$, namely, $\mathcal{H}^-(\rho(\delta_2)) = \mathcal{H}(\rho(\delta_1))$. On the other hand, it is impossible to remove any edge from $\rho(\delta_2)$ without violating the definition of route. Such a definition is in fact complete with respect to the diagnosis, but not sound in general, because of the spurious histories.

Table 1: Histories of ξ_2 and Entailed $OBS(\xi_2)$

HISTORY	OBSERVATION
$\langle\rangle$	$\langle\rangle$
$\langle T_{41}\rangle$	$\langle A_4\rangle$
$\langle T_{41}, T_{31}, T_{42}\rangle$	$\langle A_4, A_3\rangle$
$\langle T_{31}, T_{41}, T_{42}\rangle$	$\langle A_3, A_4\rangle$
$\langle T_{41}, T_{31}, T_{42}, T_{41}\rangle$	$\langle A_4, A_3, A_4\rangle$
$\langle T_{31}, T_{41}, T_{42}, T_{41}\rangle$	$\langle A_3, A_4, A_4\rangle$

If $\Delta(\xi, \xi_0) = \{\delta_1, \ldots, \delta_n\}$, the main goal of the off-line method is the generation of n diagnostic rules,

$$Rules(\xi, \xi_0) = \bigsqcup_{i=1}^{n} \{Rule(\delta_i, \mu(\delta_i))\}$$

where each rule $Rule(\delta_i, \mu(\delta_i))$ establishes an entailment from a matching condition of a (variable) observation $OBS(\xi)$ against the *matching graph* $\mu(\delta_i)$ to the diagnosis δ_i. Formally, denoting with $\Delta(OBS(\xi), \xi_0)$ the *diagnostic set* relevant to the *diagnostic problem* $(OBS(\xi), \xi_0)$, that is, the set of candidate diagnoses, and with $\asymp$ the matching relationship, the following entailment holds:

$$(OBS(\xi) \asymp \mu(\delta_i)) \models (\delta_i \in \Delta(OBS(\xi), \xi_0))$$

The matching graph $\mu(\delta_i)$ is a finite automaton obtained as a transformation[1] of the route $\rho(\delta_i)$ such that:

1. The initial state of $\mu(\delta_i)$ equals the root of $\rho(\delta_i)$;
2. Edges are labeled by observable events (messages) only (hence, owing to silent transitions, an edge may possibly be unlabeled);
3. The set of strings of messages entailed by the paths in $\mu(\delta_i)$, from the initial state to a final state, equals the set of observations entailed by the histories in $\mathcal{H}(\rho(\delta_i))$.

In other words, a matching graph $\mu(\delta_i)$ incorporates all and only the observations entailed by the histories corresponding to the diagnosis δ_i. In spite of the fact that the route $\rho(\delta_1)$ displayed on the right of Figure 2 is isomorphic to the matching graph $\mu(\delta_1)$ as well, the topology of a matching graph is in general different from any subspace of the universal space. For instance, the matching graph $\mu(\delta_2)$ relevant to the second diagnosis and depicted in Figure 3 is rather different from the corresponding route $\rho(\delta_2)$ (in fact, the universal space) displayed on the left of Figure 2.

In particular, notice that nodes are possibly duplicated (e.g., ξ_{20}), maybe with different qualification. For example, one of the two occurrences of ξ_{20} is still the initial state (even if not final, as in the corresponding route), while the other occurrence is final (but not initial). Besides, one can verify that every observation entailed by a faulty history in the universal space displayed in Figure 2 is entailed as well by a path from the initial state to a final state (called a *complete path*) in the matching graph.

[1] As detailed in Section 3.5, this transformation involves a set-theoretic difference among graphs.

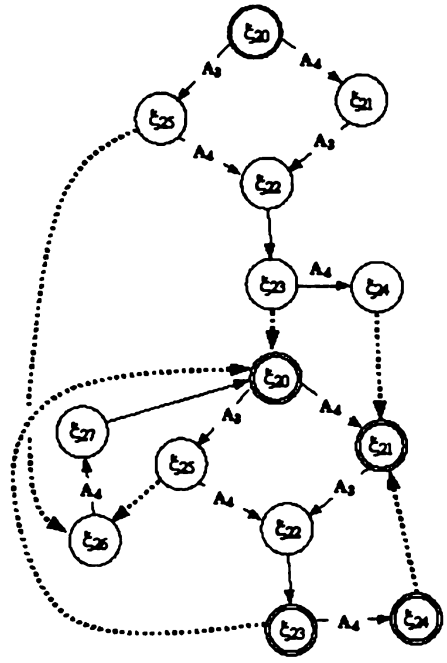


Figure 3: Matching Graph $\mu(\delta_2)$

Interestingly enough, a matching graph $\mu(\delta_i)$ can be thought of as the regular expression $\mathcal{E}$ defined on the alphabet of messages such that the whole set of strings in $\mathcal{E}$ equals the set of observations entailed by all the histories corresponding to the diagnosis δ_i. Furthermore, it is in principle possible to eliminate the nondeterminism from $\mu(\delta_i)$ so as to speed up the matching operation[2].

This way, we have classified the (possibly unbound) set of observations consistent with the histories of the universal space, namely $OBS(\mathcal{U}(\xi, \xi_0))$, so that each class (that is, each matching graph) embodies all the observations relevant to the same diagnosis, formally:

$$OBS(\mathcal{U}(\xi, \xi_0)) = \bigsqcup_{i=1}^{n} OBS(\mu(\delta_i))$$

where $OBS(\mu(\delta_i))$ denotes the set of observations entailed by all the complete paths in $\mu(\delta_i)$.

The set of diagnostic rules, $Rules(\xi, \xi_0)$, is generated off-line. Considering cluster ξ_2, two rules are obtained,

$$Rules(\xi_2, \xi_{20}) = \{Rule(\delta_1, \mu(\delta_1)), Rule(\delta_2, \mu(\delta_2))\}$$

[2] The elimination of the nondeterminism from $\mu(\delta_i)$ is *per se* not essential to the diagnostic method. A trade-off must be considered between the (off-line) computational cost of such a transformation and the advantages obtained (on-line) by the matching operation in using the deterministic automaton.

where $\delta_1 = \emptyset$, $\delta_2 = \{C_3\}$, and $\xi_{20} = (S_{31}, S_{41})$. On-line, after a reaction has finished, a diagnostic problem $(OBS(\xi_2), \xi_{20})$ has to be faced, for instance, $((\langle A_4, A_3, A_4\rangle, (S_{31}, S_{41}))$. Thus, $OBS(\xi_2)$ has to be matched against $\mu(\delta_1)$ and $\mu(\delta_2)$. In spite of the fact that different matching graphs correspond to the observations of different set of histories, the same observation may be shared among several matching graphs. That is, in general, even if $i \neq j$, it may be $OBS(\mu(\delta_i)) \cap OBS(\mu(\delta_j)) \neq \emptyset$.

For example, $OBS(\xi_2) = \langle A_4, A_3, A_4\rangle$ both matches $\mu(\delta_1)$ and $\mu(\delta_2)$. In fact, such an observation is entailed by $h(\xi_2) = \langle T_{41}, T_{31}, T_{42}, T_{41}\rangle$, $h'(\xi_2) = \langle T_{41}, T_{31}, T_{42}, T_{32}, T_{41}\rangle$, and $h''(\xi_2) = \langle T_{41}, T_{31}, T_{42}, T_{41}, T_{32}\rangle$, where $h(\xi_2) \models \delta_1$, $h'(\xi_2) \models \delta_2$, and $h''(\xi_2) \models \delta_2$. In other words, $\langle A_4, A_3, A_4\rangle \in (OBS(\mu(\delta_1)) \cap OBS(\mu(\delta_2)))$. Generally speaking, given n matching graphs (corresponding to n different candidate diagnoses), an observation is bound to match k matching graphs, $1 \leq k \leq n$, so that the actual diagnostic set consists of the diagnoses relevant to the graphs $\mu(\delta_i)$ for which the matching is successful. Formally, the diagnostic set $\Delta(OBS(\xi), \xi_0)$ is composed as follows:

$$\{\delta_i \mid Rule(\delta_i, \mu(\delta_i)) \in Rules(\xi, \xi_0), OBS(\xi) \asymp \mu(\delta_i)\}.$$

Considering our example, we have

$$\Delta(\langle A_4, A_3, A_4\rangle, (S_{31}, S_{41})) = \{\emptyset, \{C_3\}\}.$$

3 FORMALIZATION

3.1 NOTATIONS

Notation 1 (*constructors*) Let $E_1, \ldots, E_n$ be a group of elements. $(E_1, \ldots E_n)$ is a *tuple*, $\{E_1, \ldots, E_n\}$ a *set*, and $\langle E_1, \ldots E_n\rangle$ an ordered set with repetition or *sequence*. An *empty* set or sequence is either denoted by$\{\}$, $\langle\rangle$, respectively, or by the polymorphic symbol $\emptyset$.

Notation 2 (*subsequence*) Let Q and Q' be two sequences. Q' is said to be a *subsequence* of Q, denoted by $Q' \subseteq Q$, if $\forall q \in Q'$ $(q \in Q)$, and the relative order among elements in Q' is the same as in Q.

Notation 3 (*tuple selector*) Let $\tau = (E_1, \ldots, E_k)$ be a tuple. $E_i(\tau)$, $i \in \{1 \cdots k\}$, denotes the i-th element of τ.

Notation 4 (*cardinality*) Let $\mathcal{Z}$ be either a set or sequence. The number of elements in $\mathcal{Z}$, namely, the *cardinality* of $\mathcal{Z}$, is denoted by $|\mathcal{Z}|$.

Notation 5 (*composition*) If $Q = \langle E_1, \ldots, E_n\rangle$ and $Q' = \langle E'_1, \ldots, E'_m\rangle$ are two sequences, then

$$Q'' = Q \oplus Q' = \langle E_1, \ldots, E_n, E'_1, \ldots, E'_m\rangle$$

is the *composition* of Q and Q'.

Notation 6 (*permutation set*) Let $\Sigma = \{E_1, \ldots, E_n\}$ be a set. The *permutation set* of Σ, denoted by $Perm(\Sigma)$, is the whole set of sequences where each sequence is composed of all and only the elements in Σ.

Notation 7 (*ultra-powerset*) Let $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ be a set. The *ultra-powerset* of Σ, denoted by $2^{\langle\Sigma\rangle}$, is the whole set of sets of sequences defined as follows:

$$2^{\langle\Sigma\rangle} = \{\Sigma' \mid \Sigma' = \{Q_1, \ldots, Q_m\}, 2^\Sigma = \{S_1, \ldots, S_m\}, \forall i \in \{1 \cdots m\}(Q_i \in Perm(S_i))\}.$$

Notation 8 (*linearization*) Let e be a regular expression [Hopcroft and Ullman, 1979], and $\mathcal{E} = (e_1|e_2|\ldots|e_n)$ the regular expression obtained from e by recursively applying the following transformations:

1. $e'(e'_1|e'_2|\cdots|e'_k)$ is replaced by $(e'e'_1|e'e'_2|\cdots|e'e'_k)$;
2. $(e'_1|e'_2|\cdots|e'_k)e'$ is replaced by $(e'_1e'|e'_2e'|\cdots|e'_ke')$;
3. $(e'_1|e'_2|\cdots|e'_k)^*$ is replaced by $(e''_1|e''_2|\cdots|e''_p)$, where $\{e''_1, e''_2, \ldots, e''_p\} \in 2^{\langle\{e'_1, e'_2, \ldots, e'_k\}\rangle}$.

A *linearization* of e, denoted by $2^{\|e\|}$, is the set of elements in $\mathcal{E}$, namely $2^{\|e\|} = \{e_1, e_2, \ldots, e_n\}$.

3.2 MODELS

Definition 1 (*component model*) A *component model* M_C is a communicating automaton,

$$M_C = (\mathcal{S}, \mathcal{E}_{in}, \mathcal{I}, \mathcal{E}_{out}, \mathcal{O}, \mathcal{T})$$

where $\mathcal{S}$ is the set of states, $\mathcal{E}_{in}$ the set of inputs, $\mathcal{I}$ the set of input terminals, $\mathcal{E}_{out}$ the set of outputs, $\mathcal{O}$ the set of output terminals, and $\mathcal{T}$ the nondeterministic transition function, $\mathcal{T} : \mathcal{S} \times \mathcal{E}_{in} \times \mathcal{I} \times 2^{\mathcal{E}_{out} \times \mathcal{O}} \mapsto 2^{\mathcal{S}}$. A *component* is the instantiation of a component model.

Note 1.1 An input (output) event is a pair (E, ϑ), where E is an input (output) and ϑ an input (output) terminal. A transition T from state S_1 to state S_2 is triggered by an input event $\alpha = (E, I)$ at an input terminal I and generates the (possibly empty) set of output events $\beta = \{(E_1, O_1), \ldots, (E_n, O_n)\}$ at output terminals $O_1 \cdots O_n$, respectively. This is denoted by $T = S_1 \xrightarrow{\alpha|\beta} S_2$.

Definition 2 (*link model*) A *link model* M_L is a 5-tuple, $M_L = (Y, I, O, B, S)$, where $Y \in \{ASYNC, SYNC\}$ is the *type*, I the input terminal, O the output terminal, B the *capacity*, and S the *saturation mode*. The capacity B is the maximum number of events that can be buffered within the link. When the number of events in L equals B, L is *saturated*. The saturation mode of L dictates the semantics for the triggering of a transition T that generates a new output event (E, O) when L is saturated. A *link* is the instantiation of a link model.

Definition 3 (*cluster*) A *cluster* $\xi = (\mathcal{C}, \mathcal{L})$ is a connected graph where nodes are terminals of components in $\mathcal{C}$ and edges are the elements in $\mathcal{L}$, that is, the whole set of links among such terminals.

3.3 OBSERVATIONS

Definition 4 (*observation*) Let $\mathcal{P}(\xi) = \{C_1, \ldots, C_n\}$ be a partition of the components in a cluster ξ, called the *ordering partition* of ξ. An *observation item* $obs(C_i)$ of a set of components $C_i \in \mathcal{P}(\xi)$ is a temporal, totally-ordered sequence of messages relevant to components in C_i, namely $obs(C_i) = \langle m_1, \ldots, m_p \rangle$. An *observation* $OBS(\xi)$ of ξ is the tuple composed of all the observation items $obs(C_i)$, $OBS(\xi) = (obs(C_1), \ldots, obs(C_n))$. The symbol *ukn* denotes the unknown observation. An *index* K of an observation $OBS(\xi)$ is the tuple of integer variables, $K = (k_1, \ldots, k_n)$, where each k_i corresponds to $C_i \in \mathcal{P}(\xi)$, such that $k_i \in \{0, 1, \ldots, |obs(C_i)|\}$. When $\forall i \in \{1 \cdots n\}$ $(k_i = |obs(C_i)|)$, K is *complete*.

Definition 5 (*observation restriction*) The *restriction* of an observation item $obs(C_i)$ on a set of components $C_i' \subseteq C_i$, $C_i' \neq \emptyset$, denoted by $obs_{\langle C_i' \rangle}(C_i)$, is the temporal, totally-ordered sequence of messages in $obs(C_i)$ relevant to components in C_i'. Likewise, the restriction of an observation $OBS(\xi) = (obs(C_1), \ldots, obs(C_n))$ on a non-empty set C' of components in ξ is defined as follows: $OBS_{\langle C' \rangle}(\xi) = (obs_{\langle C' \rangle}(C_1), \ldots, obs_{\langle C' \rangle}(C_n))$.

Definition 6 (*observation extension*) Let $OBS(\xi) = (obs(C_1), \ldots, obs(C_n))$ be a cluster observation. The *extension* of $OBS(\xi)$, denoted by $\|OBS(\xi)\|$, is the finite set of sequences of messages defined as follows: if $\forall obs(C) \in OBS(\xi)(obs(C) = \langle\rangle)$ then $\|OBS(\xi)\| = \{\langle\rangle\}$, otherwise $\|OBS(\xi)\| = \{\omega_1, \ldots, \omega_p\}$, where each $\omega = \langle m_1, \ldots, m_q \rangle \in \|OBS(\xi)\|$, $q = \sum_{i=1}^{n} |obs(C_i)|$ is such that $\forall i \in \{1 \ldots n\}$ $(\omega_{\langle C_i \rangle} = obs(C_i))$.

3.4 DIAGNOSES

Definition 7 (*diagnostic problem*) A pair $\wp(\xi) = (OBS(\xi), \xi_0)$, where $OBS(\xi)$ is an observation of cluster ξ with initial state ξ_0, is called a *diagnostic problem* for ξ. When $OBS(\xi) = ukn$, $\wp(\xi)$ is called a *universal* diagnostic problem for ξ.

Definition 8 (*active space*) Let $\xi = (\mathcal{C}, \mathcal{L})$ be a cluster, $\mathcal{C} = \{C_1, \ldots, C_p\}$, $OBS(\xi)$ a relevant observation, ξ_0 the initial state of ξ, and $M_i = (\mathcal{S}_i, \mathcal{E}_{in_i}, \mathcal{I}_i, \mathcal{E}_{out_i}, \mathcal{O}_i, \mathcal{T}_i)$ the model of the generic component $C_i \in \mathcal{C}$, $i \in \{1 \cdots p\}$. The *active space* of the diagnostic problem $\wp(\xi) = (OBS(\xi), \xi_0)$, $Act(\wp(\xi))$, is a finite automaton $(\mathcal{S}, \mathcal{E}, \mathcal{T}, S_0, \mathcal{S}_f)$, where $\mathcal{S}$, is the set of states, $\mathcal{S} \subseteq \mathcal{S}_1 \times \cdots \times \mathcal{S}_p$, $\mathcal{E}$ the set of events,

$$\mathcal{E} \subseteq \bigcup_{j=1}^{p} \left| \left(\mathcal{T}_1' \times \cdots \times \mathcal{T}_j'\right), (\mathcal{T}_1', \ldots, \mathcal{T}_j') \in 2^{\{\mathcal{T}_1, \ldots, \mathcal{T}_p\}}, \right.$$

$\mathcal{T} = \mathcal{T}^a \cup \mathcal{T}^s$ the transition function, where $\mathcal{T}^a$ is the *asynchronous* transition function while $\mathcal{T}^s$ the *synchronous* transition function, $\mathcal{T}^a \cap \mathcal{T}^s = \emptyset$, $\mathcal{T} : \mathcal{S} \times \mathcal{E} \mapsto \mathcal{S}$, S_0 the initial state, and $\mathcal{S}_f$ the set of final states, $\mathcal{S}_f \subseteq \mathcal{S}$, such that every path $S_0 \rightsquigarrow S_f$, $S_f \in \mathcal{S}_f$, called a *history* of ξ, entails $OBS(\xi)$. An active space of a universal diagnostic problem (ukn, ξ_0) is called a *universal space* of (ξ, ξ_0), namely $\mathcal{U}(\xi, \xi_0) \equiv Act(ukn, \xi_0)$.

Definition 9 (*diagnosis*) Let $\wp(\xi) = (OBS(\xi), \xi_0)$ be a diagnostic problem and $Act(\wp(\xi))$ the relevant active space. A *diagnosis* δ of $\wp(\xi)$ is the set of faulty components relevant to a history h in $Act(\wp(\xi))$. The *diagnostic set* of $\wp(\xi)$, namely $\Delta(\wp(\xi)) = \{\delta_1, \ldots, \delta_n\}$, is the whole set of diagnoses relevant to the histories in $Act(\wp(\xi))$. The *universal diagnostic set* of (ξ, ξ_0), namely $\Delta(\xi, \xi_0)$, is the diagnostic set of the universal diagnostic problem (ukn, ξ_0).

Note 9.1 If a diagnosis δ is relevant to a history h, h is said to *entail* δ, and is denoted by $h \models \delta$. Furthermore, by definition, h does not entail any other diagnosis other than δ, namely

$$\forall \delta' \neq \delta,\ \delta' \in \Delta(OBS(\xi), \xi_0)\ (h \not\models \delta').$$

Definition 10 (*diagnosis restriction*) The *restriction* $\delta_{\langle \xi \rangle}$ of a diagnosis δ on a cluster $\xi = (\mathcal{C}, \mathcal{L})$ is the set of components $\delta \cap \mathcal{C}$. If $\Delta = \{\delta_1, \ldots, \delta_n\}$ is a set of diagnoses, then $\Delta_{\langle \xi \rangle} \equiv \{\delta_{1\langle \xi \rangle}, \ldots, \delta_{n\langle \xi \rangle}\}$ (where duplicates are removed).

3.5 KNOWLEDGE COMPILATION

Definition 11 (*diagnostic expression*) Let $\wp(\xi) = (OBS(\xi), \xi_0)$ be a diagnostic problem and $Act(\wp(\xi)) =$

$(\mathcal{S}, \mathcal{E}, \mathcal{T}, S_0, \mathcal{S}_f)$ the relevant active space. The *diagnostic graph* of $\wp(\xi)$, $\Delta^*_{\mathcal{G}}(\wp(\xi)) = (\mathcal{S}, \widetilde{\mathcal{E}}, \widetilde{\mathcal{T}}, S_0, \mathcal{S}_f)$, is a finite automaton where

$$\widetilde{\mathcal{E}} = \{\tau \mid \tau \in \mathcal{E}, T \in \tau, T = S_1 \xrightarrow{\alpha|\beta} S_2, (\mathcal{F}, Flt) \in \beta\} \cup \{\emptyset\}$$

is the set of events, and $\widetilde{\mathcal{T}} : \mathcal{S} \times \widetilde{\mathcal{E}} \mapsto 2^{\mathcal{S}}$ is the transition function obtained from $\mathcal{T}$ as follows:

1. $\forall (S \xrightarrow{\tau} S' \in \mathcal{T}, \tau \in \widetilde{\mathcal{E}})\ (S \xrightarrow{\tau} S' \in \widetilde{\mathcal{T}})$;
2. $\forall (S \xrightarrow{\tau} S' \in \mathcal{T}, \tau \notin \widetilde{\mathcal{E}})\ (S \xrightarrow{\emptyset} S' \in \widetilde{\mathcal{T}})$.

The *diagnostic expression* of $\wp(\xi)$, $\Delta^*(\wp(\xi))$, is the regular expression[3] corresponding to the automaton $\Delta^*_{\mathcal{G}}(\wp(\xi))$. The *universal diagnostic graph* of (ξ, ξ_0), $\Delta^*_{\mathcal{G}}(\xi, \xi_0)$, is the diagnostic graph of the universal diagnostic problem (ukn, ξ_0). The *universal diagnostic expression* of (ξ, ξ_0), $\Delta^*(\xi, \xi_0)$, is the regular expression corresponding to the universal diagnostic graph $\Delta^*_{\mathcal{G}}(\xi, \xi_0)$.

Note 11.1 Let $\wp(\xi)$ be a diagnostic problem and $A = Act(\wp(\xi))$ the relevant active space. An element e of the linearization of $\Delta^*(\wp(\xi))$, $e \in 2^{\|\Delta^*(\wp(\xi))\|}$, is said to *entail* a diagnosis $\delta \in \Delta(\wp(\xi))$, denoted by $e \models \delta$, if

$$\delta = \{C \mid \tau \in e, T = S_1 \xrightarrow{\alpha|\beta} S_2, T \in \tau, (\mathcal{F}, Flt) \in \beta, T \text{ is relevant to component } C\}.$$

Proposition 1 *Let $\ell = 2^{\|\Delta^*(\wp(\xi))\|}$ be a linearization of the diagnostic expression $\Delta^*(\wp(\xi))$. The diagnostic set of $\wp(\xi)$ can be obtained as follows:*

$$\Delta(\wp(\xi)) = \{\delta \mid \delta = \{C \mid e \in \ell, e \models \delta\}\}.$$

Definition 12 (*subspace*) Let A be an active space. A *subspace* of A is a connected subgraph of A which consists of all the nodes and edges in A that are relevant to some of the histories in A.

Definition 13 (*linear subspace*) Let $\wp(\xi)$ be a diagnostic problem and $A = Act(\wp(\xi))$ the corresponding active space. The *linear subspace* of A relevant to $\{e_1, \ldots, e_k\} \subseteq 2^{\|\Delta^*(\wp(\xi))\|}$, denoted by $Sub(A, \{e_1, \ldots, e_k\})$, is the smallest subspace of A such that $\forall i \in \{1 \cdots k\}$ there exists a history $h_i \in Sub(A, \{e_1, \ldots, e_k\})$ such that e_i is a subsequence of h_i.

[3] An algorithm for generating a regular expression corresponding to the language of all the possible strings of labels associated with the transitions in a finite automaton can be found in [Hopcroft and Ullman, 1979]. Thus, it is always possible to generate a diagnostic expression $\Delta^*(\wp(\xi))$ from the diagnostic graph $\Delta^*_{\mathcal{G}}(\wp(\xi))$.

Definition 14 (*route*) Let $Act(\wp(\xi))$ be the active space of a diagnostic $\wp(\xi)$ and δ a diagnosis in $\Delta(\wp(\xi))$. A *route* $\rho(\delta)$ is the smallest subspace of $Act(\wp(\xi))$ such that:

1. $\rho(\delta)$ is rooted in ξ_0;
2. Each history in $Act(\wp(\xi))$ entailing δ is also a history in $\rho(\delta)$.

The set of histories in $\rho(\delta)$ which entail δ is denoted by $\mathcal{H}(\delta)$.

Proposition 2 *Let $Act(\wp(\xi))$ be the active space of a diagnostic problem $\wp(\xi)$, δ a diagnosis in $\Delta(\wp(\xi))$, $\mathcal{E} = \{e \mid e \in 2^{\|\Delta^*(\wp(\xi))\|}, e \models \delta\}$ the subset of the linearization of the corresponding diagnostic expression entailing δ, and $\rho(\delta)$ a route in A. Then,*

$$Sub(A, \mathcal{E}) = \rho(\delta).$$

Definition 15 (*diagnostic space*) Let $\rho(\delta) = (\mathcal{S}, \mathcal{E}, \mathcal{T}, S_0, \mathcal{S}_f)$ be a route, and $\lambda = \{\ell_1, \ldots, \ell_k, \ldots\}$ a countable set of labels. A *diagnostic space* γ of δ is an automaton, namely, $\gamma(\delta) = (\mathcal{S}_\gamma, \mathcal{E}_\gamma, \mathcal{T}_\gamma, S_{0_\gamma}, \mathcal{S}_{f_\gamma})$, where $\mathcal{S}_\gamma \subseteq \lambda \times \mathcal{S}$ is the set of states, $\mathcal{E}_\gamma = \mathcal{E}$ the set of events, $\mathcal{T}_\gamma = \mathcal{T}$ the transition function, $S_0^\gamma = S_0$ the initial state, and $\mathcal{S}_f^\gamma \subseteq \mathcal{S}^\gamma$ the set of final states, such that:

1. $\forall S_{0_\gamma} \rightsquigarrow S_f$ in $\gamma(\delta), S_f \in \mathcal{S}_{f_\gamma}\ (S_0 \rightsquigarrow S_f \in \mathcal{H}(\delta))$;
2. $\forall S_0 \rightsquigarrow S_f \in \mathcal{H}(\delta)\ (\exists S_{0_\gamma} \rightsquigarrow S_f \text{ in } \gamma(\delta) \mid S_f \in \mathcal{S}_{f_\gamma})$.

Definition 16 (*stratification hierarchy*) Let $\wp(\xi)$ be a diagnostic problem and $\Delta(\wp(\xi)) = \{\delta_1, \ldots, \delta_n\}$ the corresponding diagnostic set. The *stratification hierarchy* of $\Delta(\wp(\xi))$, $\nabla(\wp(\xi)) = (\mathcal{N}, \mathcal{E}, \mathcal{N}_0)$, is a (possibly unconnected) directed acyclic graph where $\mathcal{N} = \{\delta_1, \ldots, \delta_n\}$ is the set of nodes, $\mathcal{E} : \mathcal{N} \mapsto 2^{\mathcal{N}}$ the set of edges, and $\mathcal{N}_0 \subseteq \mathcal{N}$, the set of *roots*, such that, denoting with $Succ(\delta)$ the set of successive nodes of δ, that is, $Succ(\delta) = \{\delta' \mid \delta' \in \mathcal{N}, \delta \mapsto \delta' \in \mathcal{E}\}$, the following conditions hold:

1. $\forall \delta \in \mathcal{N}\ (Succ(\delta) = \{\delta' \mid \delta' \subset \delta\})$;
2. $\forall \delta \in \mathcal{N}, \forall \delta' \in Succ(\delta), \forall \delta'' \in Succ(\delta), \delta' \neq \delta''\ (\delta' \not\subset \delta'', \delta'' \not\subset \delta')$;
3. $\forall \delta \in \mathcal{N}_0\ (\nexists \delta' \in \mathcal{N} \mid \delta \subset \delta')$.

The *universal stratification hierarchy* of (ξ, ξ_0), $\nabla(\xi, \xi_0)$, is the stratification hierarchy of the universal diagnostic set $\Delta(\xi, \xi_0)$.

Proposition 3 *Let $\nabla(\wp(\xi)) = (\mathcal{N}, \mathcal{E}, \mathcal{N}_0)$ be a stratification hierarchy, $\delta \in \mathcal{N}$ a diagnosis, $\rho(\delta)$ a route in $Act(\wp(\xi))$, and $Succ(\delta) = \{\delta_1, \ldots, \delta_k\}$ the successive nodes of δ in $\nabla(\wp(\xi))$. Then, the diagnostic space $\gamma(\delta)$ can be computed by the following set-theoretic expression:*

$$\gamma(\delta) = \rho(\delta) - \Big|_{i=1}^{k} \rho(\delta_i). \quad (1)$$

Note 16.1 Let $\wp(\xi) = (OBS(\xi), \xi_0)$ be a diagnostic problem. Exploiting (1), it is possible to generate a set of diagnostic spaces $\Gamma(\wp(\xi)) = \{\gamma(\delta_1), \ldots, \gamma(\delta_n)\}$, called the *diagnostic partition* of $\wp(\xi)$, each diagnostic space relevant to a diagnosis in $\Delta(\wp(\xi)) = \{\delta_1, \ldots, \delta_n\}$. The *universal diagnostic partition* of (ξ, ξ_0), $\Gamma(\xi, \xi_0)$, is the diagnostic partition of the universal diagnostic problem (ukn, ξ_0).

Definition 17 (*matching graph*) Let $\gamma(\delta) = (\mathcal{S}, \mathcal{E}, \mathcal{T}, S_0, S_f)$, $\mathcal{T} = \mathcal{T}^a \cup \mathcal{T}^s$, be the diagnostic space relevant to a diagnosis δ in the universal diagnostic set $\Delta(\xi, \xi_0)$. Let $\mu(\delta) = (\mathcal{S}, \widetilde{\mathcal{E}}_\mu, \widetilde{\mathcal{T}}_\mu, S_0, S_f)$ be a finite automaton, where $\widetilde{\mathcal{E}}_\mu = 2^{\mathcal{E}_{obs}}$ is the set of events such that $\mathcal{E}_{obs}$ is the domain of messages involved in the models of components in ξ, and $\widetilde{\mathcal{T}}_\mu : \mathcal{S}_\mu \times \widetilde{\mathcal{E}}_\mu \mapsto 2^{\mathcal{S}}$ is the transition function obtained from $\mathcal{T}$ as follows:

1. $\forall (S \xrightarrow{E} S' \in \mathcal{T}^a,\ E = S_1 \xrightarrow{\alpha|\beta} S_2,\ (m, Msg) \in \beta)$ $(S \xrightarrow{\{m\}} S' \in \widetilde{\mathcal{T}}_\mu)$;
2. $\forall (S \xrightarrow{E} S' \in \mathcal{T}^s,\ E = \{T_1, \ldots, T_k\},\ E_\mu = \{T'_1, \ldots, T'_{k'}\}$ is the subset of observable transitions of E, $E_\mu \neq \emptyset$, $\forall i \in \{1 \cdots k'\}(T'_i = S_1 \xrightarrow{\alpha|\beta} S_2,$ $(m'_i, Msg) \in \beta))$ $(S \xrightarrow{\{m'_1, \ldots, m'_{k'}\}} S' \in \widetilde{\mathcal{T}}_\mu)$;
3. $\forall$ silent transition $S \xrightarrow{E} S' \in \mathcal{T}$ $(S \xrightarrow{\emptyset} S' \in \widetilde{\mathcal{T}}_\mu)$.

$\mu(\delta)$ is called the *matching graph* for diagnosis δ. The *matching expression* $\mu^*(\delta)$ is the regular expression corresponding to the finite automaton $\mu(\delta)$. A string $\Omega = \langle \omega_1, \ldots, \omega_k \rangle \in \mu^*(\delta)$ is a *matching string* of $\mu^*(\delta)$.

Note 17.1 Due to synchronism, each symbol of a string in $\mu^*(\delta)$ consists in general of a set of messages. That is, a string in $\mu^*(\delta)$ is a sequence of set of messages relevant to components in ξ. In practice, it is possible to test whether an observation $OBS(\xi)$ matches $\mu^*(\delta)$ by means of the matching graph $\mu(\delta)$.

Note 17.2 A matching string $\Omega = \langle \omega_1, \ldots, \omega_r \rangle \in \mu^*(\delta)$ implicitly defines a finite set $\|\Omega\|$ of sequences of messages, called the *extension* of Ω, that is, if $\Omega = \langle\rangle$ then $\|\Omega\| = \{\langle\rangle\}$, otherwise $\|\Omega\| = \{Q \mid Q = q_1 \oplus \cdots \oplus q_r, \forall i \in \{1 \cdots r\}(q_i \in Perm(\omega_i))\}$.

3.6 KNOWLEDGE EXPLOITATION

Definition 18 (*match*) Let $OBS(\xi) = (obs(C_1), \ldots, obs(C_n))$ be an observation of cluster ξ and $\mu(\delta) = (\mathcal{S}, \mathcal{E}, \mathcal{T}, S_0, S_f)$ the matching graph for δ. The observation $OBS(\xi)$ is said to *match* $\mu(\delta)$, denoted by $OBS(\xi) \asymp \mu(\delta)$, if and only if $\exists \Omega \in \mu(\delta)$ such that $\|\Omega\| \cap \|OBS(\xi)\| \neq \emptyset$.

Proposition 4 *Let $\mathcal{U}(\xi, \xi_0)$ be a universal space, $\mu(\delta)$ the matching graph relevant to $\delta \in \Delta(\xi, \xi_0)$, and $OBS(\xi)$ an observation. If $OBS(\xi) \asymp \mu(\delta)$ then $\delta \in \Delta(OBS(\xi), \xi_0)$.*

Definition 19 (*diagnostic rule*) Let $\mathcal{U}(\xi, \xi_0)$ be a universal space. A *diagnostic rule*, $Rule(\delta, \mu(\delta))$, is an association between a diagnosis $\delta \in \Delta(\xi, \xi_0)$ and a matching graph $\mu(\delta)$. The *rule set* of (ξ, ξ_0) is the whole set of diagnostic rules relevant to $\Delta(\xi, \xi_0)$, that is, $Rules(\xi, \xi_0) = \{Rule(\delta, \mu(\delta)) \mid \delta \in \Delta(\xi, \xi_0)\}$. Let $Rules(\xi, \xi_0)$ be a rule set and $OBS(\xi)$ an observation. The *matching rule set* of the diagnostic problem $(OBS(\xi), \xi_0)$ is the subset of $Rules(\xi, \xi_0)$ defined as follows: $Matching(OBS(\xi), \xi_0) = \{R \mid R \in Rules(\xi, \xi_0), R = Rule(\delta, \mu(\delta)), OBS(\xi) \asymp \mu(\delta)\}$.

Proposition 5 *Let $\wp(\xi)$ be a diagnostic problem. The diagnostic set of $\wp(\xi)$ equals the set of diagnoses relevant to the matching rule set of $\wp(\xi)$, that is,*

$$\Delta(\wp(\xi)) = \{\delta \mid Rule(\delta, \mu(\delta)) \in Matching(\wp(\xi))\}.$$

4 ALGORITHMS

Algorithm 1 (*Compile*) The *Compile* function is a high level description of the method of knowledge compilation detailed in Section 3.5, which is capable of extracting the whole set of diagnostic rules inherent to a given universal space.

function *Compile*($\mathcal{U}(\xi, \xi_0)$): the rule set $Rules(\xi, \xi_0)$
input
$\mathcal{U}(\xi, \xi_0) = (\mathcal{S}, \mathcal{E}, \mathcal{T}, S_0, S_f)$: a universal space;
begin{Compile}
$\Delta^*(\xi, \xi_0)$:= the universal diag. expression of (ξ, ξ_0);
ℓ := a linearization $2^{\|\Delta^*(\xi, \xi_0)\|}$;
$\Delta(\xi, \xi_0) := \{\delta \mid \delta = \{C \mid e \in \ell, \tau \in e, T \in \tau,$
$T = S_1 \xrightarrow{\alpha|\beta} S_2, (\mathcal{F}, Flt) \in \beta, T$ relevant to comp. $C\}\}$;
$\mathcal{P}(\ell) := \{\ell_1, \ldots, \ell_n\}$ = a partition of ℓ such that
$\forall i \in \{1 \cdots n\}(\forall e \in \ell_i\ (e \models \delta_i, \delta_i \in \Delta(\xi, \xi_0))$;
ρ^* := the set of routes

$\{\rho(\delta_i) \mid \rho(\delta_i) = Sub(\mathcal{U}(\xi,\xi_0),\ell_i), \ell_i \in \mathcal{P}(\ell)\}$;
$\nabla(\xi,\xi_0) = (\mathcal{N},\mathcal{E},\mathcal{N}_0) :=$
the universal diagnostic hierarchy of $\Delta(\xi,\xi_0)$;
$\Gamma(\xi,\xi_0)$:= the universal diagnostic partition of (ξ,ξ_0),
$\{\gamma(\delta) \mid \delta \in \mathcal{N}, \gamma(\delta) = \rho(\delta) - \bigcup_{\delta' \in Succ(\delta)} \rho(\delta')\}$;
$\mu^* := \{\mu(\delta) \mid \mu(\delta)$ is a matching graph, $\delta \in \Delta(\xi,\xi_0)\}$;
$\mathcal{R} := \{R \mid R = Rule(\delta,\mu(\delta)), \delta \in \Delta(\xi,\xi_0), \mu(\delta) \in \mu^*\}$;
return $\mathcal{R}$
end{Compile}.

Proposition 6 *The* Compile *algorithm is a sound and complete implementation of the* Rules *operator, that is,* $Compile(\mathcal{U}(\xi,\xi_0)) = Rules(\xi,\xi_0)$.

Algorithm 2 (*Match*) The *Match* function implements the technique for performing the match defined in Section 3.6, which is used on-line in order to check a given cluster observation against the matching graph of a diagnostic rule.

```
function Match(OBS(ξ), μ(δ)): boolean
 input
  OBS(ξ): an observation of cluster ξ,
  μ(δ): a matching graph;

 function Recurse(N, OBS(ξ), K): boolean
  input
   N: a node of the matching graph μ(δ),
   OBS(ξ) = (obs(C_1), ..., obs(C_n)): an observation,
   K: the index of OBS(ξ);
  begin {Recurse}
   if N ∈ S_f(μ(δ)) and K is complete then
    return true;
   for each edge N --ω--> N' in μ(δ) do
    begin
     ω' := ω;
     if ω' = ∅ then
      if Recurse(N', OBS(ξ), K) then
       return true;
     K' := K;
     for i := 1 to n do
      if K'[i] < |obs(C_i)| then
       begin
        m := obs(C_i)[K'[i] + 1];
        while m ∈ ω' and ω' ≠ ∅ do
         begin
          ω' := ω' - {m};
          K'[i] := K'[i] + 1;
          if K'[i] < |obs(C_i)| then
           m := obs(C_i)[K'[i] + 1]
          else break
         end;
        if ω' = ∅ then
         if Recurse(N', OBS(ξ), K') then
          return true
       end;
   return false
  end {Recurse};

 begin {Match}
  S_0 := the root of μ(δ);
  return Recurse(S_0, OBS(ξ), (0, 0, ..., 0))
 end {Match}.
```

Proposition 7 *The* Match *algorithm is a sound and complete implementation of the* $\asymp$ *matching operator, that is,* $Match(OBS(\xi),\mu(\delta))$ *iff* $OBS(\xi) \asymp \mu(\delta)$.

5 DISCUSSION

Two families of approaches to diagnosis of DESs are described in the literature: model-based and chronicle-based. The former are very interesting from a knowledge acquisition point of view but their performances are limited by the computational cost of on-line abductive reasoning. The latter, instead, perform on-line rule checking, where each rule associates a set of candidate diagnoses with a sequence (or, more generically, a temporal pattern) of observed events, namely a *chronicle*. The algorithmic complexity of chronicle recognition is linear with the number of chronicles, therefore, chronicle-based approaches are more efficient than model-based approaches. However, acquisition, validation, and maintenance of knowledge concerning chronicles is extremely hard, if not impossible.

Recently, there have been several attempts to combine the efficiency of on-line rule-checking with the knowledge-related advantages of model-based diagnosis. The result of one of these attempts is the *global diagnoser* approach [Sampath et al., 1995, Sampath et al., 1996], which is further extended in [Chen and Provan, 1997]. According to this approach, the complete behavioral models of system components, each represented as an automaton, are processed off-line in order to produce a global system model. The global system model is in turn exploited for the off-line creation of another automaton, the diagnoser. At run time, a monitoring module checks the system observation against the diagnoser.

The global diagnoser approach is appealing in that every piece of observation can be handled in a constant time during on-line monitoring. However, the need for a global system model and then a global diagnoser significantly reduces the advantages of compositional modeling and makes the approach not applicable in practice to large systems, since it runs into significant computational difficulties, as pointed out in [Rozé, 1997].

Another attempt to bridge rule-based and model-based diagnosis of DESs is automatic *chronicle generation* [Laborie and Krivine, 1997] starting from the system model, which is a set of nondeterministic automata with timed transitions. However, this method raises the problems of modeling the behavior of the whole system and of exhaustively simulating it.

The approach briefly described in this paper is a further attempt in the same direction. Our method, similarly to [Sampath et al., 1995, Sampath et al., 1996], features compositional modeling but, unlike it, does not need the creation of any global system model. The core of the off-line activities is the construction of a graph representing all the possible evolutions of the system over time, starting from a given initial state and ending in a quiescent state. This makes our approach resemble [Laborie and Krivine, 1997], since behavior construction is indeed a model simulation. However, in contrast with the chronicle generation approach, the simulation performed by our approach is atemporal. Besides, simulation further differentiates our method from the global diagnoser approach, which does not include any simulation since the global model is itself a representation of all the possible system behaviors over time. This is a consequence of the fact that the global diagnoser approach allows for a smaller class of systems than ours, as the former considers synchronous systems only while the latter allows for systems exhibiting both synchronous and asynchronous behavior. Thus, the global diagnoser approach performs an off-line synchronous composition of component models into the global system model, then it accomplishes an abductive reasoning for compiling the global system model in order to obtain the diagnoser. Our approach, instead, performs a simulation, which includes both synchronous and asynchronous composition steps and enforces the constraint that the final state of any system history be quiescent. Afterwards, it performs off-line abductive reasoning while drawing matching graphs from universal spaces.

In order to reduce the computational effort, behavior construction may be inherent to subsystems instead of the whole system. Besides, any behavior construction problem can be decomposed into a multi-level hierarchy of subproblems, where each subproblem is solved separately and independent subproblems can possibly be solved in parallel [Lamperti and Zanella, 1999].

Also the nature of the on-line results of the global diagnoser method differ from those of our approach. In fact, the former method is primarily meant to monitoring the system, then it supplies as output both a set of states, which includes the current state of the system, and a (possibly ambiguous) set of faults for every state. Our approach, instead, is primarily meant to diagnose a system once its reaction to an external event has finished and then it provides as output the set of candidate diagnoses. Indeed, the universal spaces created off-line enable also on-line monitoring (this topic is however beyond the scope of the paper).

6 CONCLUSION

This paper deals with diagnosis of DESs, focusing on off-line model-based reasoning in order to compile the structural and behavioral models of the physical system at hand so as to automatically generate general diagnostic rules for it. In front of a specific diagnostic problem, such rules can be exploited on-line so as to produce the candidate diagnoses, in a rule-based fashion. This method, encompassing computationally costly off-line activities and efficient on-line activities, is an answer to possible needs for a fixed response time of the diagnosis task. When such needs are immaterial, a method that has already been presented in the literature can alternatively be adopted [Baroni et al., 1999, Lamperti and Zanella, 1999], which is based on the same modeling primitives and features on-line activities only.

Indeed the two methods, *off-line diagnosis* and *on-line diagnosis*, are interoperable, in the sense that they can be integrated within a single diagnostic procedure, even though, in principle, each of them can be used on its own. Specifically, first of all, the system has to be decomposed into sub-systems, or *clusters*, then a set of diagnostic rules can be generated, possibly in parallel, for each cluster in an arbitrary set Ξ. Cluster model compilation is organized within a hierarchy of sub-clusters, so as to increase efficiency and provide potential for better reuse.

Upon the completion of the system reaction to an external event, application-dependent criteria establish the focus of attention for diagnosis, that is, the subsystem σ to be diagnosed. For example, the reaction to a short circuit by the protection apparatus of a power transmission network causes the isolation of a sub-network on which attention is focused, which is displayed in real-time by the supervision software to the operators. Once σ has been identified, two cases are possible, corresponding to the fact that σ is contained in (or equal to) a cluster ξ for which a set of diagnostic rules are available, or not. In the first case, the set of candidate diagnoses is determined by checking the observation of σ against the diagnostic rules of ξ and then by projecting the diagnoses relevant to ξ on the components of σ. Otherwise, a *coverage* $\Sigma \subseteq \Xi$ of σ must be determined, this being the (possibly empty) set of clusters in Ξ which intersect σ, and, then, on-line diagnosis is performed, based on the observation and the active spaces of the clusters in Σ, generated off-line while determining the diagnostic rules.

In any case, the set of candidate diagnoses of a given diagnostic problem inherent to a specific cluster σ is

complete but not sound. In fact, all the diagnoses are determined which are consistent with the given observation and, in the first case, with the topology of cluster $\xi' = \xi$, or, in the second case, with the topology of the cluster $\xi' = \Sigma \cup \sigma$. Therefore, one or several candidate diagnoses may be physically impossible, as the constraints relevant to the events exchanged between cluster ξ' and the remainder of the physical system are not enforced.

Notice that, the integrated approach either reduces to off-line or on-line diagnosis if $\sigma \in \Xi$ or $\Xi = \emptyset$, respectively.

The adopted concept of observation is very general. Thus, checking at runtime whether an observation matches a graph μ of a given rule does not simply amount to checking whether the observation is a phrase of the grammar represented by μ, inasmuch as an observation is not a single phrase, as is assumed in [Sampath et al., 1995, Sampath et al., 1996, Laborie and Krivine, 1997], instead it is a set of phrases, each pertaining to a distinct group of components. Symbols within each phrase are temporally ordered, while the temporal order among the symbols within distinct phrases is unknown. Then, in principle, every possible permutation of the symbols should be considered, provided that it fulfills the ordering of the symbols belonging to each single phrase. However, an efficient algorithm, which is sound and complete, has been envisaged, so as to check the actual observation against matching graphs without performing any permutation generation.

Finally, it is worth underlying that each of the three approaches, namely, on-line, off-line, and integrated, are scalable to real-size systems and amenable to parallel and distributed computation.

References

Baroni, P., Lamperti, G., Pogliano, P., Tornielli, G., and Zanella, M. (1997). A multi-interpretation approach to fault diagnosis in power transmission networks. In *IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, 961–966, Hull, UK.

Baroni, P., Lamperti, G., Pogliano, P., and Zanella, M. (1998). Diagnosis of active systems. In *ECAI-98*, 274–278, Brighton, UK.

Baroni, P., Lamperti, G., Pogliano, P., and Zanella, M. (1999). Diagnosis of large active systems. *Artificial Intelligence*, 110(1):135–183.

Brand, D. and Zafiropulo, P. (1983). On communicating finite-state machines. *Journal of ACM*, 30(2):323–342.

Chen, Y. and Provan, G. (1997). Modeling and diagnosis of timed discrete event systems - a factory automation example. In *American Control Conference*, 31–36, Albuquerque, NM.

Cordier, M. and Thiébaux, S. (1994). Event-based diagnosis for evolutive systems. In *Fifth International Workshop on Principles of Diagnosis*, 64–69, New Paltz, NY.

Hopcroft, J. and Ullman, J. (1979). *Introduction to Automata Theory*. Addison-Wesley, Reading, MA.

Laborie, P. and Krivine, J. (1997). Automatic generation of chronicles and its application to alarm processing in power distribution systems. In *Eighth International Workshop on Principles of Diagnosis*, Mont St. Michel, F.

Lamperti, G. and Pogliano, P. (1997). Event-based reasoning for short circuit diagnosis in power transmission networks. In *IJCAI-97*, 446–451, Nagoya, J.

Lamperti, G., Pogliano, P., and Zanella, M. (1999). Modular diagnosis of distributed discrete-event systems by incremental clusterization techniques. In *International Conference on Control and Modeling of Complex Systems*, Samara, Russia.

Lamperti, G., Pogliano, P., and Zanella, M. (2000). Diagnosis of active systems by automata-based reasoning techniques. *Applied Intelligence*. To appear.

Lamperti, G. and Zanella, M. (1999). Diagnosis of discrete-event systems integrating synchronous and asynchronous behavior. In *Tenth International Workshop on Principles of Diagnosis*, 129–139, Loch Awe, UK.

Rozé, L. (1997). Supervision of telecommunication network: a diagnoser approach. In *Eighth International Workshop on Principles of Diagnosis*, Mont St. Michel, F.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. (1995). Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. (1996). Failure diagnosis using discrete-event models. *IEEE Transactions on Control Systems Technology*, 4(2):105–124.

An Algorithm for Belief Revision

Renata Wassermann
Institute for Logic, Language and Computation
University of Amsterdam
Plantage Muidergracht 24
1018-TV, Amsterdam, The Netherlands
email: renata@wins.uva.nl

Abstract

In this paper we show that a particular construction of belief revision operator is equivalent to the standard method for computing consistency-based diagnosis. We show how a diagnosis problem can be translated into a problem of belief revision and show how kernel constructions for revision operators can be used for computing diagnosis. We also show how Reiter's algorithm for computing diagnosis can be adapted for being used in belief revision.

1 Introduction

Belief revision (for an overview, see [Gärdenfors, 1988; Gärdenfors and Rott, 1995]) deals with the problem of how to accommodate new assertions into an existent body of knowledge. Traditionally, the body of knowledge is represented by a belief set, a set of formulas closed under logical implication.

Instead of belief sets we are going to use belief bases to represent belief states. A belief base is a set not closed under logical consequence [Fuhrmann, 1991; Hansson, 1989; Nebel, 1992]. For every belief base B, its closure $Cn(B)$ is a belief set that represents the beliefs held by the agent. The elements of B are assumed to be in a sense more basic beliefs, from which the elements of $Cn(B) \setminus B$ are derived. Belief bases have substantial advantages in terms of computability [Nebel, 1998], and their increased expressive power as compared to belief sets can be used to represent important features of actual belief systems [Hansson, 1992].

In AGM theory [Alchourrón *et al.*, 1985], three forms of belief change are identified: contraction, expansion, and revision. Contraction consists of retracting a specified sentence from the belief set. Expansion consists of adding a specified sentence to the belief set. If the old and the new information are not logically compatible, then the new belief state after expansion will be inconsistent. Revision is consistency-preserving incorporation of new information, i.e. if the input sentence is consistent, then the new belief set will be consistent. If necessary, consistency is obtained by deleting parts of the original belief set. These operations have also been defined for belief bases [Fuhrmann, 1991; Hansson, 1991; Hansson, 1999b].

Two additional operations of change on belief bases were introduced in [Hansson, 1991] and [Hansson, 1997]: consolidation and semi-revision. Consolidation consists in making an inconsistent belief base consistent. Semi-revision is an operation that may either accept or reject the input sentence. (For corresponding operations on belief sets, see [Makinson, 1997] and [Hansson, 1999a]).

Expansion is the only of the five operations which does not involve any extra-logical information. Expanding a belief set K by α consists in taking the logical consequences of K together with α, i.e., $K + \alpha = Cn(K \cup \{\alpha\})$. The result of the expansion of a belief base B by α is simply the union of B and α, i.e., $B + \alpha = B \cup \{\alpha\}$.

The other four operations are not uniquely defined and have been characterized in the literature by means of rationality postulates and constructions. The main purpose of this paper is to show that one sort of construction found in the literature is directly related to the constructive approach given by Reiter for finding diagnoses in faulty systems [Reiter, 1987].

We will show how a diagnosis problem can be translated into an operation of kernel semi-revision. Kernel semi-revision consists in adding new information to a database and restoring consistency if necessary. To restore consistency, the expanded database is contracted by $\perp$.

We will show how the traditional algorithm for consistency-based diagnosis given by Reiter can be used for implementing semi-revision.

Winslett suggests the use of belief revision techniques for modeling diagnosis, but without analyzing the similarities between the constructions proposed in both fields [Winslett, 1995]. She only shows how a particular problem of diagnosis can be formalized as a belief revision problem. Nebel also points that syntax-based approaches to belief revision are appropriated for problems of diagnosis, but without exploring the similarity of the constructions [Nebel, 1998].

Beyond just reducing the diagnosis problem to a problem of belief revision, the present paper aims at opening a cross-fertilization process between two communities. Researchers working on belief revision rely on very elegant and precise logical formalisms, but are very far from implementing a realistic belief revisioner. On the other hand, researchers working in the field of diagnosis have very powerful tools to prune the computational complexity of the problem, allowing them to deal with real-world situations. But several applications lack a clear formalization. By showing that, at least at a high level, the problems are equivalent, we claim that techniques developed by the model-based diagnosis community could be used for implementing belief revision.

In the rest of this paper we consider L to be a propositional language closed under the usual truth-functional connectives and containing a constant $\bot$ denoting falsum.

2 Constructions for Belief Revision

Partial meet contraction, introduced in [Alchourrón *et al.*, 1985], uses a selection function to select some of the maximal subsets of a belief set (or belief base) which do not imply the formula to be contracted. The result of the contraction is the intersection of the selected subsets. There are two limiting cases: only one subset is selected or all subsets are selected. In the first case, the operation is called a maxichoice contraction and in the second, a full meet contraction.

Hansson introduced another construction for contraction operators, called *kernel contraction* [Hansson, 1994], which is a generalization of the operation of safe contraction defined in [Alchourrón and Makinson, 1985]. The idea behind kernel contraction is that, if we remove from the belief base B at least one element of each α-kernel (minimal subset of B that implies α), then we obtain a belief base that does not imply α [Hansson, 1994]. To perform these removals of elements, we use an incision function, i.e., a function that selects at least one sentence from each kernel.

Definition 2.1 *[Hansson, 1994] The* **kernel** *operation $\perp\!\!\!\perp$ is the operation such that for every set B of formulas and every formula α, $X \in B \perp\!\!\!\perp \alpha$ if and only if:*

1. $X \subseteq B$
2. $\alpha \in Cn(X)$
3. *for all Y, if $Y \subset X$ then $\alpha \notin Cn(Y)$*

The elements of $B \perp\!\!\!\perp \alpha$ are called α-kernels.

Definition 2.2 *[Hansson, 1994] An* **incision function** *for B is any function σ such that for any formula α:*

1. $\sigma(B \perp\!\!\!\perp \alpha) \subseteq \bigcup(B \perp\!\!\!\perp \alpha)$, *and*
2. *If $\emptyset \neq X \in B \perp\!\!\!\perp \alpha$, then $X \cap \sigma(B \perp\!\!\!\perp \alpha) \neq \emptyset$.*

Semi-revision consists of two steps: first the belief α is added to the base, and then the resulting base is consolidated, i.e., contracted by $\bot$.

Definition 2.3 *[Hansson, 1997] The* **kernel semi-revision** *of B based on an incision function σ is the operator $?_\sigma$ such that for all sentences α:*

$B?_\sigma\alpha = (B \cup \{\alpha\}) \setminus \sigma((B \cup \{\alpha\}) \perp\!\!\!\perp \bot)$

Theorem 2.4 *[Hansson, 1997] An operator $?$ is an operator of kernel semi-revision if and only if for all sets B of sentences:*

- $\bot \notin Cn(B?\alpha)$ *(consistency)*
- $B?\alpha \subseteq B \cup \{\alpha\}$ *(inclusion)*
- *If $\beta \in B \setminus B?\alpha$, then there is some $B' \subseteq B \cup \{\alpha\}$ such that $\bot \notin Cn(B')$ and $\bot \in Cn(B' \cup \{\beta\})$ (core-retainment)*
- $(B + \alpha)?\alpha = B?\alpha$ *(pre-expansion)*
- *If $\alpha, \beta \in B$, then $B?\alpha = B?\beta$ (internal exchange)*

Kernel operations are more general than partial meet, i.e., all partial meet operations can be obtained by kernel operations but the converse does not hold [Hansson, 1999b].

3 Consistency-Based Diagnosis

Diagnosis is a very active area within the artificial intelligence community. The problem of diagnosis consists in, given an observation of an abnormal behavior, finding the components of the system that may have caused the abnormality [Reiter, 1987].

In the area known as model-based diagnosis [Hamscher *et al.*, 1992], a model of the device to be diagnosed is given in some formal language. In this paper, we will concentrate on model-based diagnosis methods that work by trying to restore the consistency of the system description and the observations.

In this section we introduce the standard method for calculating consistency-based diagnosis, presented in [Reiter, 1987]. Although Reiter's framework is based on first-order logic, most of the problems studied in the literature do not make use of full first-order logic and can be easily represented in a propositional language. For the sake of simplicity, we will adapt the definitions given in [Reiter, 1987] to only mention formulas in the propositional language L.

3.1 Basic Definitions

The systems to be diagnosed will be described by a set of propositional formulas. For each component X of the system, we use a propositional variable of the form okX to indicate whether the component is working as it should. If there is no evidence that the system is not working, we can assume that variables of the form okX are true.

Definition 3.1 *A* **system** *is a pair (SD,ASS), where:*

1. *SD, the system description, is a finite set of formulas of L and*
2. *ASS, the set of assumables, is a finite set of propositional variables of the form okX.*

An **observation** is a formula of L. We will sometimes represent a system by (SD,ASS,OBS), where OBS is an observation for the system (SD,ASS).

The need for a diagnosis arises when an abnormal behavior is observed, i.e., when SD∪ASS∪{OBS} is inconsistent. A diagnosis is a minimal set of assumables that must be negated in order to restore consistency.

Definition 3.2 *A* **diagnosis** *for (SD,ASS,OBS) is a minimal set $\Delta \subseteq ASS$ such that:*

$SD \cup \{OBS\} \cup ASS \backslash \Delta \cup \{\neg okX | okX \in \Delta\}$ is consistent.

A diagnosis for a system does not always exist:

Proposition 3.3 *[Reiter, 1987] A diagnosis exists for (SD,ASS,OBS) if and only if SD∪{OBS} is consistent.*

Definition 3.2 can be simplified as follows:

Proposition 3.4 *[Reiter, 1987] The set $\Delta \subseteq ASS$ is a diagnosis for (SD,ASS,OBS) if and only if Δ is a minimal set such that $SD \cup \{OBS\} \cup (ASS \backslash \Delta)$ is consistent.*

3.2 Computing Diagnoses

In this section we will present Reiter's construction for finding diagnoses. Reiter's method for computing diagnosis makes use of the concepts of *conflict sets* and *hitting sets*. A conflict set is a set of assumables that cannot be all true given the observation:

Definition 3.5 *[Reiter, 1987] A* **conflict set** *for (SD,ASS,OBS) is a set $Conf = \{okX_1, okX_2, ..., okX_n\} \subseteq ASS$ such that $SD \cup \{OBS\} \cup Conf$ is inconsistent.*

From Proposition 3.4 and Definition 3.5 it follows that $\Delta \subseteq$ASS is a diagnosis for (SD,ASS,OBS) if and only if Δ is a minimal set such that ASS$\backslash\Delta$ is not a conflict set for (SD,ASS,OBS).

A hitting set for a collection of sets is a set that intersects all sets of the collection:

Definition 3.6 *[Reiter, 1987] Let C be a collection of sets. A* **hitting set** *for C is a set $H \subseteq \bigcup_{S \in C} S$ such that for every $S \in C$, $H \cap S$ is nonempty. A hitting set for C is minimal if and only if no proper subset of it is a hitting set for C.*

The following theorem presents a constructive approach for finding diagnoses:

Theorem 3.7 *[Reiter, 1987] $\Delta \subseteq ASS$ is a diagnosis for (SD,ASS,OBS) if and only if Δ is a minimal hitting set for the collection of minimal conflict sets for (SD,ASS,OBS).*

Example 1: Consider the circuit in Figure 1. The system description of this circuit is given by (SD,ASS), where:

$$\text{ASS} = \{okX, okY, okZ\}$$

$$\begin{aligned}\text{SD} = \; & \{(A \wedge B) \wedge okX \rightarrow D,\\ & \neg(A \wedge B) \wedge okX \rightarrow \neg D,\\ & C \wedge okY \rightarrow \neg E, \neg C \wedge okY \rightarrow E,\\ & (D \vee E) \wedge okZ \rightarrow F,\\ & \neg(D \vee E) \wedge okZ \rightarrow \neg F\}\end{aligned}$$

Suppose we have OBS=$\neg C \wedge \neg F$. This observation is inconsistent with SD $\cup$ ASS. There is only one minimal conflict set for (SD,ASS,OBS): $\{okY, okZ\}$. There are two minimal hitting sets: $\{okY\}$ and $\{okZ\}$.

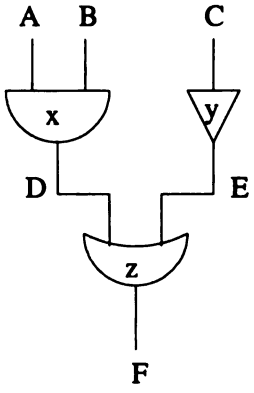


Figure 1: Circuit

4 Diagnosis via Kernel Semi-Revision

The definitions of the last section bear a striking resemblance to those of the operation of kernel semi-revision presented in Section 2.

Recall that kernel operations are based on two concepts: kernels and incision functions. The kernels are the minimal subsets of a belief base implying some sentence, while the incision functions are used to decide which elements of the kernels should be given up. Let (SD,ASS,OBS) be a system. The belief base that we are going to semi-revise corresponds to SD$\cup$ASS and the input sentence is OBS. The conflict sets are the assumables in the inconsistent kernels of SD$\cup$ASS$\cup${OBS}. So, if B=SD$\cup$ASS, the conflict sets are given by $\{X \cap \text{ASS} | X \in (B+\text{OBS}) \perp\!\!\!\perp \bot\}$. Incision functions correspond loosely to hitting sets, the minimal hitting sets being the values of minimal incisions that return only assumables. Note that there is a difference in the status of formulas in SD and those in ASS: formulas in ASS represent expectations and are more easily retracted than those in SD (cf. Definition 4.1).

We can model the diagnosis problem as a kernel semi-revision by the observation. Semi-revision can be divided in two steps. First the observation is added to the system description together with the assumables. In case the observation is consistent with the system description together with the assumables, no formula has to be given up. Otherwise, we take the inconsistent kernels and use an incision function to choose which elements of the kernels should be given up.

In the case of diagnosis, we do not wish to give up sentences belonging to the system description or the observation. We prefer to give up the formulas of the form okX, where X is a component of the system. Moreover, we are interested in minimal diagnosis, so the incision should be minimal. For this, we use a slightly different form of incision function. We modify Definition 2.2 so that incisions are minimal and elements of a given set A are prefered over the others:

Definition 4.1 *Given a set A, an* **A-minimal incision function** *is any function σ_A from sets of sets of formulas into sets of formulas such that for any set S of sets of formulas:*

1. $\sigma_A(S) \subseteq \bigcup S$,
2. *If* $\emptyset \neq X \in S$, *then* $X \cap \sigma_A(S) \neq \emptyset$,
3. *If for all* $X \in S$, $X \cap A \neq \emptyset$, *then* $\sigma_A(S) \subseteq A$, *and*
4. $\sigma_A(S)$ *is a minimal set satisfying 1,2, and 3.*

If we take A to be the set of assumables, we obtain an incision function that prefers to select formulas of the form okX over the others.

We can show that for (SD,ASS,OBS), whenever a diagnosis exists, an ASS-minimal incision function will select only elements of ASS:

Proposition 4.2 *Let (SD,ASS,OBS) be a system with an observation and σ_{ASS} an ASS-minimal incision function. If a diagnosis exists, then*

$\sigma_{ASS}((SD \cup ASS \cup \{OBS\}) \perp\!\!\!\perp \bot) \subseteq ASS$.

Proof: A diagnosis exists if and only if SD is consistent with OBS (Proposition 3.3). Hence, every inconsistent kernel of SD$\cup$ASS$\cup${OBS} must contain an element of ASS. From Definition 4.1, it follows that $\sigma_{ASS}((\text{SD} \cup \text{ASS} \cup \{\text{OBS}\}) \perp\!\!\!\perp \bot) \subseteq \text{ASS}$. □

Lemma 4.3 *The assumables that occur in an inconsistent kernel of the set SD$\cup$ASS$\cup${OBS} form a conflict set for (SD,ASS,OBS) and all minimal conflict sets can be obtained in this way, i.e.:*

(i) For every $X \in (SD \cup ASS \cup \{OBS\}) \perp\!\!\!\perp \bot$, $X \cap ASS$ is a conflict set, and

(ii) For every minimal conflict set Y, there is some set X such that $X \in (SD \cup ASS \cup \{OBS\}) \perp\!\!\!\perp \bot$ and $X \cap ASS = Y$.

Proof:

(i) Let $X \in (SD \cup ASS \cup \{OBS\}) \perp\!\!\!\perp \perp$. Then, $X \subseteq (X \cap ASS) \cup SD \cup \{OBS\}$. Since X is inconsistent, so is $(X \cap ASS) \cup SD \cup \{OBS\}$, hence $X \cap ASS$ is a conflict set.

(ii) Let Y be a minimal conflict set. Then $Y \cup SD \cup \{OBS\}$ is inconsistent and since $Y \subseteq ASS$, there is some $X \in (SD \cup ASS \cup \{OBS\}) \perp\!\!\!\perp \perp$ such that $X \cap ASS \subseteq Y$. Suppose by contradiction that there is some formula α such that $\alpha \in Y$ but $\alpha \notin X \cap ASS$. Since $X \cap ASS$ is a conflict set for (SD,ASS,OBS), this contradicts the minimality of Y. Hence, $X \cap ASS = Y$. □

Note that not every inconsistent kernel determines a minimal conflict set, since for conflict sets only the elements of ASS matter, i.e., there may be two inconsistent kernels X_1 and X_2 such that $X_1 \cap ASS$ is a proper subset of $X_2 \cap ASS$.

Recall that given an incision function σ, the semi-revision of a set B by a formula α was given by $B?_\sigma \alpha = (B + \alpha) \setminus \sigma((B + \alpha) \perp\!\!\!\perp \perp)$. A diagnosis is given by the elements of ASS that are given up in a kernel semi-revision by the observation.

Proposition 4.4 *Let S=(SD,ASS,OBS) be a system and σ_{ASS} an ASS-minimal incision function.*

$(SD \cup ASS \cup \{OBS\}) \setminus ((SD \cup ASS)?_{\sigma_{ASS}} OBS) = \sigma_{ASS}((SD \cup ASS \cup \{OBS\}) \perp\!\!\!\perp \perp)$ is a diagnosis.

Proof: We have to prove that given a system for which there is a diagnosis and an observation, it holds that:

1. If d is a diagnosis, then there is an ASS-minimal incision function σ_{ASS} such that $d = \sigma_{ASS}((SD \cup ASS \cup \{OBS\}) \perp\!\!\!\perp \perp)$.
2. If σ_{ASS} is an ASS-minimal incision function, then $\sigma_{ASS}((SD \cup ASS \cup \{OBS\}) \perp\!\!\!\perp \perp)$ is a diagnosis.

1. Let $\sigma_{ASS}((SD \cup ASS \cup \{OBS\}) \perp\!\!\!\perp \perp) = d$. We have to show that σ_{ASS} is an ASS-minimal incision function for the relevant domain, i.e., we must show that it satisfies the four conditions of Definition 4.1.

(i) $d \subseteq \bigcup((SD \cup ASS \cup \{OBS\}) \perp\!\!\!\perp \perp)$: If d is a diagnosis according to Definition 3.2, then d is a minimal hitting set for the set of all minimal conflicts of (SD,ASS,OBS). From part (ii) of Lemma 4.3 we know that for every minimal conflict set Y, there is $X \in (SD \cup ASS \cup \{OBS\}) \perp\!\!\!\perp \perp$ such that $X \cap ASS = Y$.

(ii) If $\emptyset \neq X \in (SD \cup ASS \cup \{OBS\}) \perp\!\!\!\perp \perp$, then $X \cap d \neq \emptyset$: From part (i) of Lemma 4.3, we know that $X \cap ASS$ is a conflict set. Since d is a hitting set, $X \cap d \neq \emptyset$.

(iii) If $d \not\subseteq ASS$, then $X \cap ASS = \emptyset$ for some $X \in (SD \cup ASS \cup \{OBS\}) \perp\!\!\!\perp \perp$: Since d is a diagnosis, $d \subseteq ASS$ and the condition is trivialy satisfied.

(iv) d is a minimal subset satisfying (i),(ii),(iii): Since d is a diagnosis according to Definition 3.2, d is a minimal hitting set.

2. From part (ii) of Lemma 4.3, we have that all minimal conflict sets are elements of the set $\{X \cap ASS | X \in (SD \cup ASS \cup \{OBS\}) \perp\!\!\!\perp \perp\}$. We have to show that an ASS-minimal incision function for the inconsistent kernels determines a minimal hitting set for all minimal conflicts. From Definition 4.1 and Proposition 4.2 it follows that $\sigma_{ASS}((SD \cup ASS \cup \{OBS\}) \perp\!\!\!\perp \perp)$ is a hitting set for the set of minimal conflicts of (SD,ASS,OBS). That it is also a minimal hitting set follows directly from Definition 4.1. (Since the non-minimal conflict sets contained in $\{X \cap ASS | X \in (SD \cup ASS \cup \{OBS\}) \perp\!\!\!\perp \perp\}$ are supersets of some minimal conflict set and all minimal conflict sets are considered, an ASS-minimal incision function will give the same result as if only minimal conflict sets were considered). □

Going back to the circuit in Figure 1, we see that $SD \cup ASS \cup \{OBS\}$ is inconsistent. This means that $SD \cup ASS \cup \{OBS\}$ has to be consolidated. There is only one inconsistent kernel:

$$(SD \cup ASS \cup \{OBS\}) \perp\!\!\!\perp \perp = \{\{\neg C \wedge okY \rightarrow E, (D \vee E) \wedge okZ \rightarrow F, okY, okZ, \neg C \wedge \neg F\}\}$$

We have two possibilities for ASS-minimal incision functions: $\sigma_1 = \{okY\}$ and $\sigma_2 = \{okZ\}$. This means that either Y or Z are not working well.

5 Reiter's algorithm

The algorithm given in [Reiter, 1987] computes all minimal hitting sets for an arbitrary collection of sets. We will use it later for finding the incision functions used in kernel constructions. We present here the version corrected by [Greiner *et al.*, 1989].

The algorithm generates a directed acyclic graph (DAG) with nodes labeled by sets and arcs labeled by elements of the set. The idea is that for each node labeled by a set S, the arcs leaving from it are labeled by the elements of S. Let $H(n)$ denote the set formed by the labels of the path going from the root to node n. Node n has to be labeled by a set S such that $S \cap H(n) = \emptyset$. If no such set can be found, the node is

labeled by @. The idea is that every path finishing at a node labeled by @ is a hitting set, since it intersects all possible labels for the nodes.

The algorithm tries to generate as few new node labels as possible. This is due to the fact that for diagnosis (and for belief revision as well), the collection of sets F which can be used as node labels will be given only implicitly. Calculating one element of F involves a call to a theorem prover to find a conflict set (in the case of diagnosis; a kernel in the case of belief revision) and is therefore a very expensive operation.

The algorithm minimizes the number of calls to the theorem prover by pruning the graph while it is being built. When a new node has to be labeled, the algorithm tries to re-use existing labels first. If a node label S is a superset of another label S', then it can be "closed", it does not have to be considered any longer, since any hitting set for F will be a hitting set for $F \setminus \{S\}$.

Let F be a family of sets.

1. Choose one set to label the root node (level 0).

2. For each node n at level i do:

2.a. If n is labeled by a set S, then for every $s \in S$ create an arc departing from n with label s.

2.b. Set $H(n)$ to be the set of arc labels on the path from the root to node n.

2.c. If there is some node n' such that $H(n') = H(n) \cup \{s\}$, then let the s-arc of n point to n'.

2.d. Else, if there is a node n' labeled by @ such that $H(n') \subset (H(n) \cup \{s\})$ then close the s-arc (i.e., do not compute a label or successors for this node).

2.e. Else, if there is some node n' labeled by S' such that $S' \cap (H(n) \cup \{s\}) = \emptyset$, then let the s-arc of n point to a new node labeled by S'.

2.f. Otherwise, let the s-arc point to a new node m and let m be labeled by the first element S' of F such that $S' \cap H(m) = \emptyset$. If no such set exists, then label m by @.

2.g. If there is some node n' labeled by a set S_1 such that $S' \subset S_1$, then relabel node n' by S' and remove all arcs departing from n' which were labeled by elements of $S' \setminus S_1$.

3. Repeat step 2 for level $i + 1$.

The algorithm expands the graph breadth first. Each level is processed by step 2. Steps 2.c and 2.e re-use nodes or labels if possible. Reiter has proven the following theorem:

Theorem 5.1 *[Reiter, 1987] Let F be a collection of sets and let D be a graph returned by the algorithm above. The set $\{H(n) | n$ is a node of D labeled by @$\}$ is the collection of minimal hitting sets for F.*

The final algorithm for calculating all diagnoses constructs a DAG as above, except that when it is supposed to generate a new node label, it does so by calling the theorem prover with a smaller set. Let TP be a function such that TP(SD,ASS,OBS) returns a conflict set for (SD,ASS,OBS), i.e, a subset S of ASS such that SD∪S∪{OBS} is inconsistent. If no conflict set exists, the function returns @. When one needs to compute a label for a node n, label n by TP(SD,ASS\H(n),OBS).

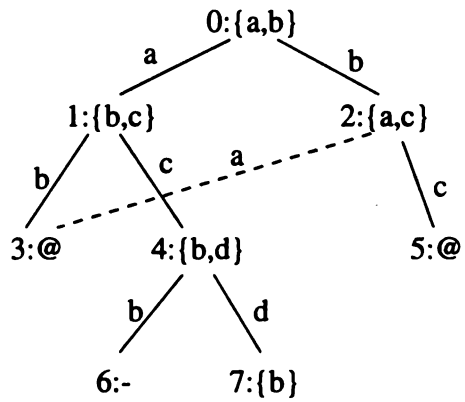


Figure 2: Reiter's algorithm – 1

Consider the following example [Greiner *et al.*, 1989]:

Example 2: Let $F = \{\{a,b\}, \{b,c\}, \{a,c\}, \{b,d\}, \{b\}\}$. Figure 2 shows part of the graph built by the algorithm. The set $\{a,b\}$ is chosen to label node 0 and two arcs are created with labels a and b. Node 1 is labeled by $\{b,c\}$ and node 2 by $\{a,c\}$ and arcs are created leaving from node 1 labeled by b and c and leaving from node 2 labeled by a and c. Node 3 receives the label @, since there is no set $S \in F$ such that $S \cap H(3) = S \cap \{a,b\} = \emptyset$. Node 4 is labeled by $\{b,d\}$. The arc leaving from node 2 and labeled by a points to node 3, since $H(3) = H(2) \cup \{a\}$ (step 2.c of the algorithm). Node 5 is labeled by @, since there is no set $S \in F$ such that $S \cap H(5) = S \cap \{b,c\} = \emptyset$. Node 6 is closed (step 2.d of the algorithm), since nodes 3 and

5 are labeled by @ and $H(3) \subset H(6)$ and $H(5) \subset H(6)$. When node 7 is labeled by $\{b\} \subset \{a, b\}$, the graph is pruned and the root node 0 is relabeled by $\{b\}$ (step 2.g). The resulting graph is shown in Figure 3. The hitting sets are $\{a, b\}$ and $\{b, c\}$.

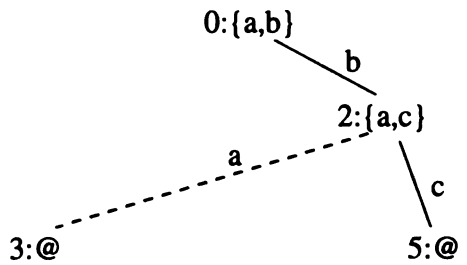


Figure 3: Reiter's algorithm – 2

6 An algorithm for kernel semi-revision

As we have seen, Reiter's algorithm computes all minimal hitting sets for an arbitrary collection of sets. We will use it later for finding the incision functions used in kernel constructions.

In order to apply the algorithm for kernel operations, one needs to adapt very few things. Usually, we will not have access to the whole collection of inconsistent kernels. Using a theorem prover in order to find an inconsistent subset of a belief base does not guarantee that the set returned is a minimal one. Nevertheless, even if the set is not minimal, the algorithm returns the collection of values for the minimal incision functions for all inconsistent subsets of the base. In particular, the returned values are values for incision functions for the inconsistent kernels.

Let TP be a function such that TP(B) returns an inconsistent subset of B. We then build a directed acyclic graph using Reiter's algorithm. Whenever a new label for a node n has to be generated, we call TP(B\H(n)).

That the algorithm does what it is expected to do follows directly from the correctness of Reiter's algorithm.

Example 3: Consider the belief base $B = \{\neg a, \neg b, a \vee b, q, \ q \rightarrow p, \neg p\}$. There are only two inconsistent kernels, $\{\neg a, \neg b, a \vee b\}$ and $\{q, q \rightarrow p, \neg p\}$. However, the theorem prover may find some superset of these sets. Suppose it finds the collection $\{\{\neg a, \neg b, a \vee b, q\}, \{\neg b, a \vee b, q, q \rightarrow p, \neg p\}, \{\neg a, \neg b, a \vee b, q, \neg p\}, \{\neg a, q, q \rightarrow p, \neg p\}, \{\neg a, \neg b, a \vee b\}, \{q, q \rightarrow p, \neg p\}\}$.

The values for the minimal incision functions are: $\{\neg a, q \rightarrow p\}$, $\{\neg a, q\}$, $\{\neg a, \neg p\}$, $\{\neg b, q \rightarrow p\}$, $\{\neg b, q\}$, $\{\neg b, \neg p\}$, $\{a \vee b, q \rightarrow p\}$, $\{a \vee b, \neg p\}$, and $\{a \vee b, \neg p\}$.

7 Conclusion and Future Work

We have translated a diagnosis problem into a problem of kernel semi-revision where the values of the incision function used must be minimal. It is not difficult to see that kernel operations where the incisions are minimal are equivalent to maxichoice operations. Maxichoice contractions have been shown to have undesirable results when applied to belief sets [Alchourrón *et al.*, 1985]. However, as was argued by Makinson in [Makinson, 1987], they are perfectly acceptable operations when applied to belief bases. This claim can be confirmed by the fact that the diagnosis community, which is not interested in closed theories, has been using maxichoice contraction for finding minimal diagnoses.

We have also shown how Reiter's algorithm for diagnosis can be adapted for implementing belief revision operators. The fact that Reiter's algorithm can be used for belief revision bridges the gap between belief revision theory and implemented systems. Reiter's algorithm is used in several systems and we expect that several computational tools developed for diagnosis systems can be adapted for revision operators.

Reiter's algorithm expands the graph breadth first, generating at the end all possible values for minimal incision functions. If the function TP is substituted by one that finds a minimal inconsistent set, then one can choose to expand depth-first, stopping when one solution was found. If some kind of ordering among the formulas is present (as entrenchment in belief revision or a priori failure probability in diagnosis), this ordering can be used to choose which branch to expand. In this way, it may be possible to obtain partial meet operations by encoding the selection function as the choice of branches to expand.

Acknowledgments

This work is supported by a grant from the Brazilian funding agency CAPES.

References

[Alchourrón and Makinson, 1985] Carlos Alchourrón and David Makinson. On the logic of theory change: Safe contraction. *Studia Logica*, 44:405–422, 1985.

[Alchourrón *et al.*, 1985] Carlos Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change. *Journal of Symbolic Logic*, 50(2):510–530, 1985.

[Fuhrmann, 1991] André Fuhrmann. Theory contraction through base contraction. *Journal of Philosophical Logic*, 20:175–203, 1991.

[Gärdenfors and Rott, 1995] Peter Gärdenfors and Hans Rott. Belief revision. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume IV, chapter 4.2. Oxford University Press, 1995.

[Gärdenfors, 1988] Peter Gärdenfors. *Knowledge in Flux - Modeling the Dynamics of Epistemic States.* MIT Press, 1988.

[Greiner *et al.*, 1989] Russell Greiner, Barbara A. Smith, and Ralph W. Wilkerson. A correction to the algorithm in Reiter's theory of diagnosis. *Artificial Intelligence*, 41(1):79–88, 1989.

[Hamscher *et al.*, 1992] Walter Hamscher, Luca Console, and Johan de Kleer, editors. *Readings in Model-Based Diagnosis.* Morgan Kaufmann, 1992.

[Hansson, 1989] Sven Ove Hansson. New operators for theory change. *Theoria*, 55:114–132, 1989.

[Hansson, 1991] Sven Ove Hansson. *Belief Base Dynamics.* PhD thesis, Uppsala University, 1991.

[Hansson, 1992] Sven Ove Hansson. In defense of base contraction. *Synthese*, 91:239–245, 1992.

[Hansson, 1994] Sven Ove Hansson. Kernel contraction. *Journal of Symbolic Logic*, 59:845–859, 1994.

[Hansson, 1997] Sven Ove Hansson. Semi-revision. *Journal of Applied Non-Classical Logic*, 7(1-2):151–175, 1997.

[Hansson, 1999a] Sven Ove Hansson. A survey of non-prioritized belief revision. *Erkenntnis*, 50(2/3):413–427, 1999.

[Hansson, 1999b] Sven Ove Hansson. *A Textbook of Belief Dynamics.* Kluwer Academic Press, 1999.

[Makinson, 1987] David Makinson. On the status of the postulate of recovery in the logic of theory change. *Journal of Philosophical Logic*, 16:383–394, 1987.

[Makinson, 1997] David Makinson. Screened revision. *Theoria*, 63:14–23, 1997.

[Nebel, 1992] Bernhard Nebel. Syntax based approaches to belief revision. In Peter Gärdenfors, editor, *Belief Revision.* Cambridge University Press, 1992.

[Nebel, 1998] Bernhard Nebel. How hard is it to revise a belief base? In D. Dubois and H. Prade, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems, Vol. 3: Belief Change*, pages 77–145. Kluwer Academic, 1998.

[Reiter, 1987] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987. Reprinted in [Hamscher *et al.*, 1992].

[Winslett, 1995] Marianne Winslett. Epistemic aspects of databases. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume IV, chapter 4.2, pages 133–174. Oxford University Press, 1995.

Temporal Reasoning I

Two Problems with Reasoning and Acting in Time

Haythem O. Ismail and Stuart C. Shapiro
Department of Computer Science and Engineering
and Center for Cognitive Science
State University of New York at Buffalo
226 Bell Hall
Buffalo, NY 14260-2000
{hismail|shapiro}@cse.buffalo.edu

Abstract

Natural language competent embodied cognitive agents should satisfy two requirements. First, they should act in and reason about a changing world, using reasoning in the service of acting and acting in the service of reasoning. Second, they should be able to communicate their beliefs, and report their past, ongoing, and future actions in natural language. This requires a representation of time using a deictic NOW, that models the compositional semantic properties of the English "now". Two problems emerge for an agent that interleaves reasoning and acting in a personal time. The first concerns the representation of plans and reactive rules involving reasoning about "future NOWs". The second emerges when, in the course of reasoning about NOW, the reasoning process itself results in NOW changing. We propose solutions for the two problems and conclude that: (i) for embodied cognitive agents, time is not just the object of reasoning, but is embedded in the reasoning process itself; and (ii) at any time, there is a partonomy of NOWs representing the agent's sense of the current time at different levels of granularity.

1 Introduction: *In* and *About* Time

Reasoning about time is something that agents acting in the world ought to be capable of doing. Performing an act *before* another, achieving states that must hold *while* an act is being performed, or reasoning about states that should hold *after* the performance of an act involve, whether implicitly or explicitly, some degree of reasoning about time. Temporal logics are used for reasoning about time and discussing its properties in a precise and explicit manner (van Benthem, 1983, for instance). In these logics, time just happens to be the subject matter of some of its sentences. Except for the presence of terms, predicates, and operators denoting temporal entities and relations; there is nothing about the language that is intrinsically temporal. For example, the logics developed in (van Benthem, 1983) might be applied to one-dimensional space, the rational numbers, or the integers just by changing the denotation of some symbols. Being about time is an extrinsic property of a logic; it only determines the domain of interpretation, maybe the syntax, but not the interpretation and reasoning processes. More specifically, let Δ be a collection of logical formulas (i.e., a knowledge base) and let $\mathcal{A}$ be an acting and reasoning system reasoning with the information in Δ. In particular, think of $\mathcal{A}$ as an embodied cognitive agent acting in the world and of Δ as the contents of its memory. $\mathcal{A}$ is said to be reasoning *about* time if the semantics of some of the sentences in Δ refer to temporal individuals and properties.[1] The assumption here is that this is an accidental situation; the design of the inference rules used by $\mathcal{A}$ is tuned only to the syntax (the domain in which inference takes place) and the semantics of the logical connectives and operators. The semantics of functional terms and predicates, and the reasoning being about time has no effect on how $\mathcal{A}$'s inference engine operates.

Not only may reasoning be about time, it could also be in time. What does reasoning *in* time mean? In the technical sense in which we want to interpret "in" and in the context of Δ and $\mathcal{A}$ from above, it means two things.

1. Temporal progression is represented in Δ. That is, at any point, there is a term in Δ which, for the agent $\mathcal{A}$, denotes the current time. Which

[1] Of course, this is a very liberal characterization of what reasoning about time is.

term denotes the current time depends on *when* one inspects Δ.[2] This gives the agent "a personal sense of time" (Shapiro, 1998, p. 141).

2. Reasoning takes time. By that we do not simply mean the obvious fact that any process happens over an interval of time. What we mean is that it happens over an interval of $\mathcal{A}$'s time. In other words, the term in Δ denoting the current time at the beginning of the reasoning process is different from that denoting the current time at the end.

As we shall argue below, if one were to take the issue of reasoning and acting in time seriously, problems immediately emerge. We are going to present two problems that naturally arise when dealing with a cognitive agent reasoning and acting in certain situations. These are, by no means, unrealistic or exotic situations; they involve simple acting rules and behaviors that agents are expected to be able to exhibit. The main point is that, when it comes to embodied cognitive agents, time is not just a possible object of reasoning; it is deeply embedded into the agent's reasoning processes.

2 The Agent

In this section, we briefly highlight certain design constraints that we impose on our developing theory of agents. Our theory is based on the GLAIR agent architecture (Hexmoor et al., 1993; Hexmoor and Shapiro, 1997). This is a layered architecture, the top layer of which is responsible for high level cognitive tasks such as reasoning and natural language understanding. This level is implemented using the SNePS knowledge representation and reasoning system (Shapiro and Rapaport, 1987; Shapiro and Rapaport, 1992; Shapiro and the SNePS Implementation Group, 1999).

We use "Cassie" as the name of our agent. Previous versions of Cassie have been discussed elsewhere (Hexmoor, 1995; Shapiro, 1989). Those are actually various incarnations of the disembodied linguistically-competent cognitive agent of the SNePS system (Shapiro and Rapaport, 1987; Shapiro and Rapaport, 1995). There are four basic requirements that we believe are reasonable for a theory of embodied cognitive agents.

R1. **Reasoning in service of acting and acting in service of reasoning.** Cassie uses reasoning in the service of acting in order to decide when, how, and/or whether to act in a certain manner. Similarly, Cassie may act in order to add a missing link to a chain of reasoning. For example, conclusions about the state of the world may be derived based, not only on pure reasoning, but also on looking, searching and performing various sensory acts. For more on this see (Kumar and Shapiro, 1994).

R2. **Memory.** Cassie has a record of what she did and of how the world evolved. A memory of the past is important for reporting to others what happened. This, as shall be seen, constrains the *form* of certain sentences in the logic.

R3. **Natural language competence.** Cassie should be capable of using natural language to interact with other agents (possibly human operators). This means that SNePS representations of the contents of Cassie's memory ought to be linguistically-motivated. By that we mean two things. First, on the technical side, the representations should be designed so that they may be produced by a natural language understanding system, and may be given as input to a natural language generator. Second, at a deeper level, the syntax of the representations and the underlying ontology should reflect their natural language (in our case, English) counterparts. In particular, we admit into the SNePS ontology anything that we can think or talk about (Shapiro and Rapaport, 1987; Shapiro and Rapaport, 1992). For a general review of linguistically-motivated knowledge representation, see (Iwańska and Shapiro, 2000).

R4. **Reasoning in time.** Cassie has a personal sense of time (Shapiro, 1998); at any point, there is a term in the logic that, for Cassie, represents the current time. To represent temporal progression, we use a deictic NOW pointer (Almeida and Shapiro, 1983; Almeida, 1995)– a meta-logical variable that assumes values from amongst the time-denoting terms in the logic.[3] We will use "*NOW" to denote the term which is the value of the meta-logical variable NOW. There are four things to note. First, NOW is not a term in the logic, just a meta-logical variable. Second, "*NOW" is not itself a fixed term in the logic; at any point, it is a shorthand for the term denoting the current time.[4] Third, to maintain a personal sense of time, the value of NOW changes

[2] Note that this means that an agent reasoning in time also reasons about time.

[3] A similar idea has been suggested by (Allen, 1983).

[4] In Kaplan's terms (Kaplan, 1979; Braun, 1995), only *contents*, not *characters*, are represented in the knowledge base.

to a new term when, and only when, Cassie acts.[5] Note that this does not preclude Cassie's learning about events that happened between or during times that once were values of NOW. Fourth, for R3, the behavior of *NOW models the compositional semantic properties of the English "now".[6] It is always interpreted as the time of the utterance (or the assertion), it cannot refer to past nor to future times (Prior, 1968; Kamp, 1971; Cresswell, 1990). Note that, given R3, Cassie essentially reasons *in* time, in the sense of Section 1.

Two incarnations of embodied Cassies have been developed based on the above requirements. In the FEVAHR project (Shapiro, 1998) Cassie played the role of a "Foveal Extra-Vehicular Activity Helper-Retriever (FEVAHR)." Cassie, the FEVAHR, was implemented on a commercial Nomad robot, including sonar, bumpers, and wheels, enhanced with a foveal vision system consisting of a pair of cameras with associated hardware and software. There have also been several software simulatated versions of the FEVAHR. Cassie, the FEVAHR, operates in a $17' \times 17'$ room containing: Cassie; Stu, a human supervisor; Bill, another human; a green robot; and three indistinguishable red robots. Cassie is always talking to either Stu or Bill—taking statements, questions, and commands from that person (all expressed in a fragment of English), and responding and reporting to that person in English. Cassie can be told, by the person addressing her, to talk to the other person, or to find, look at, go to, or follow any of the people or robots in the room. Cassie can also engage in conversations on a limited number of other topics in a fragment of English, similar to some of the conversations in (Shapiro, 1989).

A more recent incarnation of embodied Cassie is as a robot that clears a field of unexploded ordnance (UXO remediation). This Cassie has only existed as a software simulation. The UXO-Cassie exists in an area consisting of four zones: a safe zone; an operating zone that possibly contains UXOs; a drop-off zone; and a recharging zone. The UXO-Cassie contains a battery that discharges as she operates, and must be recharged in the recharge zone as soon as it reaches a low enough level. She may carry charges to use to blow up UXOs. Her task is to search the operating zone for a UXO, and either blow it up by placing a charge on it, and then going to a safe place to wait for the explosion, or pick up the UXO, take it to the drop-off zone, and leave it there. The UXO-Cassie has to interrupt what she is doing whenever the battery goes low, and any of her actions might fail. (She might drop a UXO she is trying to pick up.) She takes direction from a human operator in a fragment of English, and responds and reports to that operator. There is a large overlap in the grammars of Cassie, the FEVAHR, and the UXO-Cassie.

The requirements listed above, which we believe are quite reasonable, have certain representational and ontological impacts on the formal machinery to be employed. As we have found in our experiments with Cassie, the FEVAHR, and the UXO-Cassie, and as we shall show below, this leads to problems with reasoning with the deictic NOW. Before setting out to discuss these problems, let us first introduce the basic logical infra-structure.

3 The Ontology of a Changing World

3.1 Change

Traditionally, there have been two main models for representing change. First, there is the STRIPS model of change (Fikes and Nilsson, 1971) where, at any time t, the only propositions in the knowledge base are about those states that hold at t. When time moves, propositions are added and/or deleted to reflect the new state of the world. The main obvious problem is that an agent based on such a system does not have any memory of past situations (thus violating R2). Second, there are variants of the situation calculus (McCarthy and Hayes, 1969) where propositions are associated with indicators to when they hold. Indicators may be situations as in the situation calculus (terms denoting instantaneous snapshots of the world) or time-denoting terms (Allen, 1983; Shoham, 1987). In what follows, we shall adopt the second approach for representing change. In particular, our chronological indicators shall be taken to denote times— a decision that is rooted in R3 and R4 from Section 2.

3.2 States

So far, we have been a little sloppy with our terminology. In particular, we have been using the terms "state" and "proposition" interchangeably. First, let us briefly explain what we mean by "proposition". Propositions are entities that may be the object of Cassie's belief (or, in general, of any propositional attitude). We assume propositions to be first-class entities in our ontology. Cassie's *belief space* is essentially a set of propositions— those that she believes (Shapiro, 1993).

[5] More generally, this should happen whenever Cassie recognizes a change in the world, including changes in her own state of mind.

[6] Unlike the **now** of (Lespérance and Levesque, 1995).

Cassie's beliefs are about *states* holding over time. At any given time, a given state may either hold or not hold. The notion of states referred to here is that found in the linguistics and philosophy of language literature (Vendler, 1957; Mourelatos, 1978; Galton, 1984; Bach, 1986, for instance).[7] A particularly interesting logical property of states is homogeneity; if a state s holds over some time interval t, then it holds over all subintervals of t.

States differ as to their degree of *temporal stability*. Here, we are interested in two types of states: *eternal* and *temporary*. Eternal states are related to the eternal sentences of (Quine, 1960); they either *always* hold or *never* hold. Temporary states, on the other hand, may repetitively start and cease to hold. Examples of eternal states are expressible by sentences such as *God exists*, *Whales are fish*, or *The date of John's graduation is September 5th 2001*. Temporary states are expressed by sentences such as *The litmus paper is red*, *John is in New York*, or *John is running*.

Temporary states starting or ceasing to hold are responsible for changes in the world and, hence, need to be associated with times (see Section 3.1). This association is formally achieved by introducing a function symbol, Holds, that denotes a function from temporary states and times to propositions. Thus, Holds(s, t) denotes the proposition that state $[s]$ holds over time $[t]$.[8] Note that this is similar to the situation calculus with reified fluents.

Eternal states do not change with time and, hence, should not be associated with any particular times.[9] If anything, one would need a unary function to map eternal states into propositions. More ontological economy may be achieved though if we make some observations. First, note that, unlike temporary states, eternal states cannot start, cease, or be perceived. They may only be believed to be holding, denied to be holding, asserted to be holding, wished to be holding, etc. That is, an agent can only have propositional attitudes toward eternal states. This means that the set of eternal states is isomorphic to a subset of the set of propositions. Second, all propositions may be thought of to be about eternal states holding. For example, Hold(s, t) may be thought of as denoting the eternal state of some particular temporary state holding at some particular time. Note that this is eternal since it is either always the case or never the case. Henceforth, we shall make the assumption that eternal states are identical to propositions and will use the two terms interchangeably. In this case, we do not need any functions to map eternal states, thus simplifying our syntax.

3.3 Time

As has been hinted above, we opt for an interval-based ontology of time.[10] We introduce two functional symbols, $<$ and $\sqsubseteq$, to represent the relations of temporal precedence and temporal parthood. More precisely, $t_1 < t_2$ denotes the proposition that $[t_1]$ precedes (and is topologically disconnected from) $[t_2]$ and $t_1 \sqsubseteq t_2$ denotes the proposition that $[t_1]$ is a subinterval of $[t_2]$. Because of its homogeneity, a state will be said to hold *NOW if it holds over a super-interval of *NOW.

4 The Problem of the Unmentionable Now

4.1 The Problem

How does introducing the eternal-temporary distinction affect the reasoning process? Consider the following sentence schema:

(1) IF *ant* THEN *cq*

(1) means that if Cassie believes that *ant*[11] holds, then she may also believe that *cq* holds.[12] This works fine if *ant* and *cq* denote eternal states (for example, "IF Mammals(whales) THEN Give-Birth(whales)"). However, if, instead, they denote temporary states, we need to quantify over time; the temporary state-denoting terms by themselves do not say anything about the states holding over time. (2) captures the intended meaning: if Cassie believes that *ant* holds over time t, then she may also believe that *cq* holds over t.

(2) $\forall t$ IF Holds(*ant*, t) THEN Holds(*cq*, t)

(1) and (2) represent sentences for *pure* reasoning. Doing reasoning in the service of acting requires sentences for *practical* reasoning. In particular, let us concentrate on one kind of acting rule: rules about *when*

[7] As opposed to the states of (McDermott, 1982) and (Shanahan, 1995) which are more like time points or situations of the situation calculus.

[8] If τ is a term in the logic, we use $[\tau]$ to mean the denotation of τ.

[9] Though, with time, Cassie may revise her beliefs about eternal states (Martins and Shapiro, 1988).

[10] See (Allen, 1983) and (Shoham, 1985) for arguments in support of interval semantics.

[11] For convenience, we shall, henceforth, write p in place of $[p]$ whenever what we mean is clear from context.

[12] "may" because the rule might not fire, even if Cassie believes that *ant* holds.

Assertion	Assertion Time
(5) $\forall t$ When Holds(Sounds(alarm), t) DO Leave(building)	t_1
(6) Holds(Sounds(alarm), t_0)	t_2

Table 1: A timed sequence of assertions for the fire-alarm problem.

to act.[13] Imagine Cassie operating in a factory. One reasonable belief that she may have is that, when the fire-alarm sounds, she should leave the building. The underlying schema for such a belief is represented in (3) (Kumar and Shapiro, 1994).

(3) When *cond* DO *act*

The intended interpretation for (3) is that when Cassie comes to believe that the condition *cond* holds, she should perform the act *act*. Again, this is fine so long as *cond* denotes an eternal state. If forward inference causes both *cond* and (3) to be asserted in Cassie's belief space, she will perform *act*. What if *cond* denotes a temporary state? Obviously, we need to somehow introduce time since assertions about temporary states holding essentially involve reference to time. Following (2), one may propose the following representation.

(4) $\forall t$ When Holds(*cond*, t) DO *act*

Asserting Holds(*cond*, t_1), for some particular time t_1, (4) would be matched and Cassie would perform *act*. On the face of it, (4) looks very innocent and a straight forward extrapolation of (2). However, a closer look shows that this is, by no means, the case. Using quantification over time works well for inference since the consequent is a proposition that may just happen to be *about* time. Acting, on the other hand, takes place *in* time, resulting in an interesting problem. Table 1 represents a timed sequence of assertions entered into Cassie's belief space. The left column shows the assertion, and the right column shows Cassie's term for the time of the assertion. The problem is that t_0 in (6) may refer to a time preceding t_2 (or even t_1). That is, (6) could be an assertion about the alarm sounding at some time in the past, something that we should be capable of asserting. Nevertheless, (6) matches (5) and Cassie leaves the building —at t_2— even though there is no danger!

One problem with (5) (and generally (4)) is that nothing relates the time of performing the act to the time at which the state holds. We may attempt to revise (4) by tying the action to that time.

(7) $\forall t$ When Hold(*cond*, t) DO Perform(*act*, t)

Where Perform(*act*, t) is intended to mean that Cassie should perform *act* at time t. However, this alleged semantics of Perform is certainly ill-defined; acts may only be performed *NOW, in the present. Cassie cannot travel in time to perform *act* in the past, at a time over which (she believes that) *cond* held. The basic problem seems to be quantifying over *all* times. What we really want to say is that when the state holds *NOW, perform the act. That is,

(8) When Hold(*cond*, *NOW) DO *act*

However, we cannot mention "*NOW"; it is not itself a term in the logic (see R4 in Section 2). If we replace (5) in Table 1 with the appropriate variant of (8), "*NOW" in the left column would be just a shorthand for the term appearing in the right column, namely t_1. The assertion would, therefore, be very different from what we intended it to be.

Before presenting our solution to the problem, we first need to discuss two approaches that might seem to solve it. We shall show that, although they may appear to eliminate the problem, they actually introduce more drastic ones.

4.2 A NOW Function

The basic problem, as we have shown, is that we cannot mention NOW; there is no unique term in the logic that would, at any point, denote the current time for the agent. The existence of such a term is problematic since its semantics essentially changes with time. One way to indirectly incorporate NOW within the language is to introduce a NOW function symbol. In particular, the expression NOW(t) would mean that t denotes the current time. Thus, one may express the general acting rule as follows:

(9) $\forall t$ When (Hold(*cond*, t) $\wedge$ NOW(t)) DO *act*

This might seem to solve the problem, for it necessitates that the time at which the state holds is *a current time* (and at any time, there is a unique one). There are two problems though.

[13] By "rule" we mean a domain rule, expressed in the logical language, which Cassie might come to believe as a result of being told it in natural language. We do *not* mean a rule of inference which would be implemented in the inference engine of the knowledge representation and reasoning system.

1. NOW(t) denotes a temporary state. By its very definition, the argument of NOW needs to change to reflect the flow of time. Thus, rather than using NOW(t), we should use Holds(NOW(t), t) to express the proposition that **[t]** is the current time. This gets us back where we started, since the expression Holds(NOW(t), t) would have to replace NOW(t) in (9). An assertion of Holds(NOW(t_0), t_0) with t_0 denoting some past time will cause the agent to perform *act* when it shouldn't.

2. Suppose that, at t_1, Cassie is told that John believes that t_2 is the current time. That is, "Believe(John, NOW(t_2))" is asserted. At time t_3, the same assertion provides different information about John. In particular, it attributes to John a belief that was never intended to be asserted into Cassie's belief space. The general problem is that, as time goes by, Cassie needs to revise her beliefs. Those may be her own, or other agents', beliefs about what the current time is. In the first case, the revision may be the simple deletion of one belief and introduction of another. In the second case, however, things are much more complicated as demonstrated by the above example. It should be noted that, in any case, the very idea of Cassie *changing her mind* whenever time moves is, at best, awkward, and results in Cassie *forgetting* correct beliefs that she once had (thus violating R2).

4.3 The Assertion Time

Inspecting the second row of Table 1, one may think that part of the problem is the inequality of the time appearing in the right column (t_2) to that in the left column (t_0). Indeed, if somehow we can ensure that these two times are identical, the problem may be solved. (Kamp, 1971) proposes an ingenious mechanism for correctly modeling the compositional properties of the English "now" (namely, that it always refers to the time of the utterance even when embedded within the scope of tense operators). Basically, Kamp defines the semantic interpretation function relative to two temporal indices rather than only one as in traditional model theory. The two times may be thought of as the Reichenbachian event and speech times (Reichenbach, 1947). We shall not review Kamp's proposal here; rather, based on it, we shall introduce an approach that might seem to solve the problem.

The basic idea is to move the assertion time appearing in the right column of Table 1 to the left column. That is, to formally *stamp* each assertion with the time at which it was made. Formally, we introduce a symbol Asserted that denotes a function from propositions and times to propositions. Thus, "Asserted(p, t_a)" denotes the proposition that Cassie came to believe p at t_a. We then replace (4) by (10).

(10) $\forall t$ When Asserted(Holds(*cond*, t), t) DO *act*

That is, Cassie would only perform *act* when she comes to believe that *cond* holds, at a time at which it actually holds. This will indeed not match any assertions about past times and apparently solves the problem. However, there are at least two major problems with this proposal.

1. Introducing the assertion time results in problems with simple implications as that in (1). In particular, due to its semantics, the assertion time of the antecedent need not be that of the consequent; one may come to believe in *ant* at t_1 and infer *cq* later at t_2. The problem is that the time at which the inference is made cannot be known in advance. Essentially, this is the same problem that we started with; we only know that the inference will be made at some unmentionable future *NOW.

2. So far, we have only discussed the problem in the context of forward chaining. The same problem also emerges in some cases of backward reasoning in the service of acting. For example, Cassie might have a plan for crossing the street. Part of the plan may include a conditional act: *"If the walk-light is on, then cross the street"*. Note that this is a conditional act, one that involves two things: (i) trying to deduce whether the walk-light is on, and (ii) crossing the street or doing nothing, depending on the result of the deduction process. Evidently, to formalize the act, we have the same difficulty that we have with (4). Using the assertion time proposal, one might represent the act as shown in (11), where the act following "THEN" is to be performed if the state following "ActIf" holds.

   (11) $\forall t$ ActIf Asserted(Holds(On(walk-light), t), t) THEN Cross(street)

   However, attempting to deduce Asserted(Holds(On(walk-light), t), t) will succeed even if t matches some past time, t_0, at which it was asserted that the walk-light is on. Hence, introducing the assertion time only solves the problem with forward but not backward reasoning.

Forward(s_1)

1. Perform usual forward chaining on s_1.
2. If s_1 = Holds(s_2, *NOW) then Forward(s_2).

Backward(s_1)

1. If s_1 is eternal then perform usual backward chaining on s_1.
2. Else Backward(Holds(s_1, *NOW)).

Figure 1: Modified forward and backward chaining procedures.

Assertion		Assertion Time
(12)	When Sounds(alarm) DO Leave(building)	t_1
(13)	Holds(Sounds(alarm), t_0)	t_2
(14)	Holds(Sounds(alarm), t_3)	t_3

Table 2: Fire-alarm scenario for the modified chaining procedures.

4.4 A Solution

What is the problem? At its core, the problem is that we need to make some assertions about future acts that refer to unmentionable future *NOWs. Those *NOWs would only be known at the time of acting. Even their being *future* is not something absolute that we know about them; they are only future with respect to the assertion time. We somehow need to introduce *NOW only when it is known— at the time of acting. Our proposal is to eliminate reference to time in rules like (4) (or acts like (11) for that matter) and let the inference and acting system introduce *NOW when it is using these rules. Thus, instead of (4), we shall use (3) for both cases where *cond* is eternal or temporary.

(3) When *cond* DO *act*

Figure 1 outlines modified forward and backward chaining procedures. The input to these procedures is a state (eternal or temporary) s_1. Note that *NOW is inserted, by the procedures themselves at the time of reasoning. This guarantees picking up the appropriate *NOW. Going back to the fire-alarm example, consider the timed sequence of assertions in Table 2 (which is a variant of Table 1). As illustrated in Figure 2, asserting (13) at time t_2 does not cause Cassie to leave the building. First, note that (13) does not match (12) and hence the act of leaving the building will not be activated by step 1 of the **Forward** procedure. Second, since t_0 is not identical to *NOW (t_2), the recursive call to **Forward** in step 2 will not be performed. Thus, Cassie will, correctly, not leave the building just because she is informed that the fire-alarm sounded in the past. On the other hand, as illustrated in Figure 3, at t_3 the fire-alarm actually sounds. Still, (14) does not match (12). However, since t_3 is itself *NOW, step 2 results in **Forward** being called with "Sounds(alarm)" (which matches s_2). By step 1, of the recursive call, this will match (12) resulting in Cassie, correctly, leaving the building. Similarly, we may replace (11) by (15):

Forward(Holds(Sounds(alarm), t_0))
1. Holds(Sounds(alarm), t_0) doesn't match Sounds(alarm)
2. $t_0 \neq t_2$

Figure 2: Forward inference on (13) at t_2 does not lead to acting.

Forward(Holds(Sounds(alarm), t_3))
1. Holds(Sounds(alarm), t_3) doesn't match Sounds(alarm)
2. $t_3 = t_3$
   Forward(Sounds(alarm))
   1. Sounds(alarm) matches Sounds(alarm) so Leave(building)

Figure 3: Forward inference on (14) at t_3 does lead to acting.

(15) ActIf On(walk-light)THEN Cross(street).

If Cassie is told to perform this conditional act at t_1, the procedure **Backward** would be called with "On(walk-light)" as an argument. Since this is a temporary state, backward chaining will be performed on Holds(On(walk-light), *NOW), thus querying the knowledge base about whether the walk-light is on at t_1, the time we are interested in.

5 The Problem of the Fleeting Now

Imagine the following situation. At t_1, we tell Cassie to perform the act represented in (15). The modified backward chaining procedure initiates a deduction process for Holds(On(walk-light), t_1). Using acting in service of reasoning, Cassie decides to look toward the walk-light in order to check if it is on. In order to look, Cassie moves her head (or cameras, if you will). Since time moves whenever Cassie acts, NOW moves to a new time, t_2. Cassie notices that the walk-light is

indeed on. This sensory information is represented in the form of an assertion "Holds(On(walk-light), t_3)", where *NOW $\sqsubseteq t_3$. By the homogeneity of states, this means that "Holds(On(walk-light), *NOW)". However, *NOW is t_2, not t_1, the time that we were originally interested in. Thus, the deduction fails and Cassie does not cross the street even though the walk-light is actually on!

It should be noted that this problem is not a result of the modified inference procedures. The general problem is that the very process of reasoning (which in this case involves acting) may result in changing the state in which we are interested. We are interested in the state of the world at a specific time. Sensory acts are essentially durative and whatever observations we make would be, strictly speaking, about a different time.[14] It is in the "strictly speaking" part of this last sentence that we believe the solution to the problem lies. The following sentences could be normally uttered by a speaker of English.

(16) I am *now* sitting in my office.

(17) I *now* exercise everyday.

(18) I am *now* working on my PhD.

The word "now" in the above sentences means basically the same thing: the current time. However, there are subtle differences among the three occurrences of the word. In particular, the "now" in each case has a different *size*. The "now" in (18) is larger than that in (17) which is larger than that in (16). The same observation has been made by (Allen and Kautz, 1988, p. 253). Evidently, we conceive of "now's" at different levels of granularity. The problem outlined above really lies in our treatment of *NOW at a level of granularity that is too fine for the task Cassie is executing. We are interested in a level relative to which t_1 and t_2 would be *indistinguishable* (à la (Hobbs, 1985)).

Granularity in general, and temporal granularity in particular, has been discussed by many authors (Hobbs, 1985; Habel, 1994; Euzanet, 1995; Pianesi and Varzi, 1996; Mani, 1998). However, these approaches, though quite detailed in some cases, only provide insights into the issue; they do not represent directly implementable computational solutions. What we are going to do here is *sketch* an approach, one that we intend to further pursue and refine in future work. The main idea is to give up thinking of values of NOW as single terms. Instead, each *NOW may have a rich structure of subintervals which are themselves *NOWs at finer levels of granularity. Our approach is to think of the meta-logical variable NOW not as taking the values of time-denoting terms, but rather of totally-ordered sets of time-denoting terms. More precisely, ⟨NOW, $\sqsubseteq$⟩ is a totally ordered poset.

We can think of this poset as a stack, such that the greatest and least elements are the bottom and top of the stack elements, respectively. The symbol "*NOW" is now to be interpreted as referring to the top of the stack of NOWs. Moving from one level of granularity to another corresponds to pushing or popping the stack. In particular, to move to a finer granularity, a new term is pushed onto the stack, and thus becomes *NOW. On the other hand, to move to a coarser granularity, the stack is popped. At any level, the movement of time is represented by replacing the top of the stack with a new term. Symbolically, we represent these three operations, illustrated in Fig. 4, as: ↓NOW, ↑NOW, and ↕NOW respectively (the last one is motivated by realizing that replacement is a pop followed by a push).

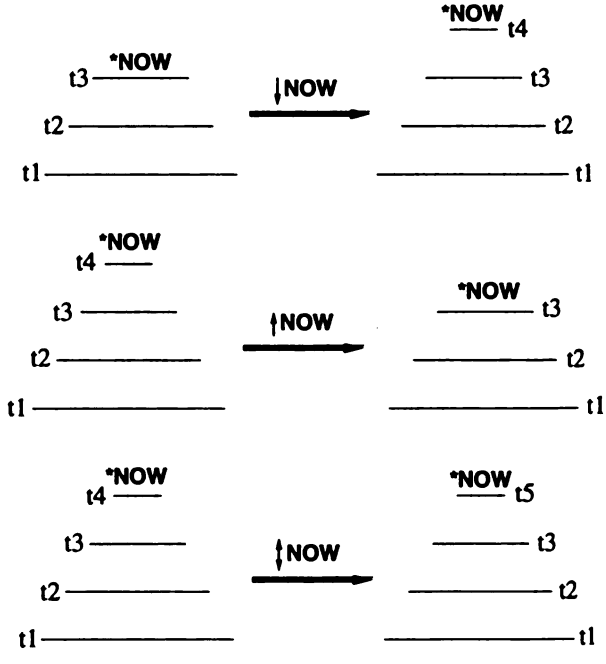


Figure 4: Operations on the stack of NOWs.

Using this mechanism, the problem outlined above may be solved as follows.

1. Cassie wonders whether "Holds(On(walk-light), t_1)".

2. Cassie decides to look towards the walk-light.

[14] Interestingly, this is the gist of the uncertainty principle in quantum physics.

3. ↓NOW (*NOW = t_2).
4. Cassie looks toward the walk-light.
5. ↕NOW (*NOW = t_3).
6. Cassie senses that the walk-light is on. That is, an assertion "Holds(On(walk-light), t_4)" is made, with $t_3 \sqsubseteq t_4$.
7. ↑NOW (*NOW = t_1).
8. Assert "$t_1 \sqsubseteq t_4$".
9. Cassie realizes that "Holds(On(walk-light), t_1)".

The above sketches a solution to the problem, it obviously does not present a complete theory. To arrive at such a theory, various questions need to be solved. First, when to push and pop the stack is still not exactly clear. Here, we decide to push into a finer granularity when acting is performed in service of reasoning (step 3). In general, one might propose to perform a push any time the achievement of a goal requires achieving sub-goals. Popping is the complementary operation, it could be performed after each sub-goal has been achieved (step 7). The most problematic step in the above solution is step 8. The motivation behind it is simple and, we believe, reasonable. When Cassie notices that the walk-light is on at t_3, it is reasonable for her to assume that it was on over a period starting before and extending over the *NOW *within* which she is checking the walk-light, namely t_1. Of course, this presupposes certain intuitions about the relative lengths of the period of the walk-light being on (t_4) and of that over which Cassie acts (t_1). In a sense, this is a variant of the frame problem (McCarthy and Hayes, 1969); given that a state s holds over interval t_1, does it also hold over a super-interval, t_2, of t_1?[15]

Our main objective here is not to provide a complete solution to the problem. Rather, we want to point the problem out, and propose some ideas about how it may be solved. Future research will consider how the proposal outlined above may be extended and refined into a concrete theory of temporal granularity that could be applied to reasoning, acting, and natural language interaction.

6 Conclusions

A reasoning, acting, natural language competent system imposes unique constraints on the knowledge representation formalism and reasoning procedures it employs. Our commitment to using a common representational formalism with such a multi-faceted system uncovers problems that are generally not encountered with other less-constrained theories. For example, a memoryless agent may use the STRIPS model of change, in which case representing temporal progression, and having to deal with the problems it raises, would not be required. A logical language that is not linguistically-motivated need not represent a notion of the present that reflects the unique semantic behavior of the natural language "now"— an issue that underlies the two problems discussed.

The problem of "the unmentionable now" results from the inability to refer to future values of the variable NOW. Since *NOW can only refer to the time of the assertion (mirroring the behavior of the English "now"), one cannot use it in the object language to refer to the future. Such reference to future now's is important for specifying conditional acts and acting rules. Our solution is to eliminate any reference to those times in the object language, but to modify the forward and backward chaining procedures so that they insert the appropriate values of NOW at the time of performing a conditional act or using an acting rule. The problem of "the fleeting now" emerges when, in the course of reasoning about (the value of) NOW, the reasoning process itself results in NOW changing. The solution that we sketched in Section 5 is based on realizing that, at any point, the value of NOW is not a single term, but rather a stack of terms. Each term in the stack corresponds to the agent's notion of the current time at a certain level of granularity, with granularity growing coarser towards the bottom of the stack. Temporal progression and granularity shifts are modeled by various stack operations.

An agent that reasons about its actions, while acting, and has a personal sense of time modeled by the relentlessly moving NOW is different in a non-trivial sense from other agents. The problem of "the unmentionable now" shows that, for such an agent, time is not just an external domain phenomenon that it, accidentally, needs to reason about. Rather, time is an internal factor that the very deep inferential processes need to take into account. The problem of "the fleeting now" moves the issue of temporal granularity from the rather abstract realm of reasoning and language understanding down to the real world of embodied acting.

7 Acknowledgments

The authors thank the members of the SNePS Research Group of the University at Buffalo for their support and comments on the work reported in this

[15] In the traditional frame problem, t_2 is a successor, not a super-interval, of t_1.

paper. This work was supported in part by ONR under contract N00014-98-C-0062.

References

Allen, J. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11).

Allen, J. F. and Kautz, H. A. (1988). A model of naive temporal reasoning. In Hobbs, J. and Moore, R., editors, *Formal Theories of the Commonsense World*, pages 261–268. Ablex Publishing Corporation, Norwood, NJ.

Almeida, M. (1995). Time in narratives. In Duchan, J., Bruder, G., and Hewitt, L., editors, *Deixis in Narrative: A Cognitive Science Approach*, pages 159–189. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ.

Almeida, M. and Shapiro, S. (1983). Reasoning about the temporal structure of narrative texts. In *Proceedings of the Fifth Annual Meeting of the Cognitive Science Society*, pages 5–9, Rochester, NY.

Bach, E. (1986). The algebra of events. *Linguistics and Philosophy*, 9:5–16.

Braun, D. (1995). What is character? *Journal of Philosophical Logic*, 24:227–240.

Cresswell, M. J. (1990). *Entities and Indices*. Kluwer Academic Publishers, Dordrecht.

Euzanet, J. (1995). An algebraic approach to granularity in qualitative time and space representation. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 894–900, San Mateo, CA. Morgan Kaufmann.

Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3/4):189–208.

Galton, A. (1984). *The Logic of Aspect*. Clarendon Press, Oxford.

Habel, C. (1994). Discreteness, finiteness, and the structure of topological spaces. In Eschenbach, C., Habel, C., and Smith, B., editors, *Topological Foundations of Cognitive Science*, pages 81–90. Papers from the Workshop at the FISI-CS, Buffalo, NY.

Hexmoor, H. (1995). *Representing and Learning Routine Activities*. PhD thesis, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY.

Hexmoor, H., Lammens, J., and Shapiro, S. C. (1993). Embodiment in GLAIR: a grounded layered architecture with integrated reasoning for autonomous agents. In Dankel, II, D. D. and Stewman, J., editors, *Proceedings of The Sixth Florida AI Research Symposium (FLAIRS 93)*, pages 325–329. The Florida AI Research Society.

Hexmoor, H. and Shapiro, S. C. (1997). Integrating skill and knowledge in expert agents. In Feltovich, P. J., Ford, K. M., and Hoffman, R. R., editors, *Expertise in Context*, pages 383–404. AAAI Press/MIT Press, Menlo Park, CA / Cambridge, MA.

Hobbs, J. (1985). Granularity. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 432–435, Los Altos, CA. Morgan Kaufmann.

Iwańska, Ł. M. and Shapiro, S. C., editors (2000). *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*. AAAI Press/The MIT Press, Menlo Park, CA.

Kamp, H. (1971). Formal properties of 'now'. *Theoria*, 37:227–273.

Kaplan, D. (1979). On the logic of demonstratives. *Journal of Philosophical Logic*, 8:81–98.

Kumar, D. and Shapiro, S. C. (1994). Acting in service of inference (and vice versa). In Dankel, II, D. D., editor, *Proceedings of the Seventh Florida Artificial Intelligence Research Symposium*, pages 207–211, St. Petersburg, FL. The Florida AI Research Society.

Lespérance, Y. and Levesque, H. (1995). Indexical knowledge and robot action– a logical account. *Artificial Intelligence*, 73(1–2):69–115.

Mani, I. (1998). A theory of granularity and its application to problems of polysemy and underspecification of meaning. In Cohn, A., Schubert, L., and Shapiro, S., editors, *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR '98)*, pages 245–255, San Francisco, CA.

Martins, J. and Shapiro, S. C. (1988). A model for belief revision. *Artificial Intelligence*, 35(1):25–79.

McCarthy, J. and Hayes, P. (1969). Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, D. and Michie, D., editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, Edinburgh, Scotland.

McDermott, D. (1982). A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101–155.

Mourelatos, A. P. D. (1978). Events, processes, and states. *Linguistics and Philosophy*, 2:415–434.

Pianesi, F. and Varzi, A. (1996). Refining temporal reference in event structures. *Notre Dame Journal of Formal Logic*, 37(1):71–83.

Prior, A. N. (1968). "Now". *Noûs*, 2(2):101–119.

Quine, W. V. O. (1960). *Word and Object*. The MIT Press, Cambridge, MA.

Reichenbach, H. (1947). *Elements of Symbolic Logic*. Macmillan Co., New York, NY.

Shanahan, M. (1995). A circumscriptive calculus of events. *Artificial Intelligence*, 77:249–284.

Shapiro, S. (1989). The CASSIE projects: An approach to natural language competence. In *Proceedings of the 4th Portugese Conference on Artificial Intelligence*, pages 362–380, Lisbon, Portugal. Springer-Verlag.

Shapiro, S. (1993). Belief spaces as sets of propositions. *Journal of Experimental and Theoretical Artificial Intelligence*, 5:225–235.

Shapiro, S. (1998). Embodied Cassie. In *Cognitive Robotics: Papers from the 1998 AAAI Fall Symposium*, pages 136–143, Menlo Park, CA. AAAI Press. Technical report FS-98-02.

Shapiro, S. and Rapaport, W. (1987). SNePS considered as a fully intensional propositional semantic network. In Cercone, N. and McCalla, G., editors, *The Knowledge Frontier*, pages 263–315. Springer-Verlag, New York.

Shapiro, S. and Rapaport, W. (1992). The SNePS family. *Computers and mathematics with applications*, 23(2–5):243–275. Reprinted in F. Lehman, ed. *Semantic Networks in Artificial Intelligence*, pages 243–275. Pergamon Press, Oxford, 1992.

Shapiro, S. and Rapaport, W. (1995). An introduction to a computational reader of narratives. In *Deixis in Narrative: A Cognitive Science Perspective*, pages 79–105. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ.

Shapiro, S. and the SNePS Implementation Group (1999). *SNePS 2.5 User's Manual*. Department of Computer Science and Engineering, State University of New York at Buffalo.

Shoham, Y. (1985). Ten requirements for a theory of change. *New Generation Computing*, 3:467–477.

Shoham, Y. (1987). Temporal logics in AI: Semantical and ontological considerations. *Artificial Intelligence*, 33:89–104.

van Benthem, J. (1983). *The Logic of Time*. D. Reidel Publisihng Company, Dordrecht, Holland.

Vendler, Z. (1957). Verbs and times. *The Philosophical Review*, 66(2):143–160.

Cyclical and Granular Time Theories as Subsets of the Herbrand Universe

Edjard Mota
e-mail: edjard@dcc.fua.br
Departamento de Ciência da Computação
ICE - Universidade do Amazonas
Av. Gen. Rodrigo Otávio, 3000 Mini-Campus
CEP 69077-000 Manaus - Brasil

Abstract

The lack of a suitable framework to the specification of simulation models of ecosystems at different grains of time, which also have cyclical events brings a new challenge to knowledge representation: how do we integrate different models with such features? We shall see in this paper how to deal with such issues by using a quantified temporal logic called **NatureTime** in which quantifiers range over moments and intervals. Here special terms of the Herbrand universe stand for temporal relations. For this we introduce an *environment* to represent temporal substitutions and a special temporal unification. The correctness of the meta-interpreter is addressed and we discuss the expressive power of the logic for dealing with models aligned and non-aligned in time.

keywords: Representation of Time, Temporal Reasoning Techniques

1 Introduction

Most work on the representation and reasoning about time use universal quantification over temporal variables for dealing with cycles and the granularity of time. This may be problematic in domains such as simulation models of ecosystems. Usually, it is not so easy for users to model cyclicity and processes interacting at many scales of time by means of an interpretation of a potentially infinite set. For "practical" reasons such models are usually built in traditional programming languages. Because of it, the behavior they are supposed to simulate is rarely a function of the meaning of the structures they manipulate. Such a functionality can be influenced only by the form of such structures, and thus we need a systematic relationship between form and meaning [Moore, 1995].

AI would contribute to simulation modeling activity by offering a special languages with the appropriate constructions to define executable models in a declarative way. To achieve this goal we have to unpack some useful knowledge from code of simulation models, and so we need a framework with a clear relation between notation and semantics. As a consequence, this knowledge can be easily accessible to modelers. Logic has all this features as was shown in the Eco-Logic project [Robertson et al., 1991]. This project was a first attempt to explore extensively the use of logic as a high level description tool for improving users' comprehension of simulation models, but with few concern with the diagnosis of such models. For this, time representation and inference mechanisms are the key points. Consider the following scenario as a motivation.

"Two models of trees, t_1 and t_2, expressed on a weekly time scale. Both influence each other and their attributes being updated at different times, and they also interact with a model of an insect pest, say b_1. It moves up and down on a daily basis, reversing direction when it reaches the top or bottom. It goes away during the raining season from December up to April."

The temporal knowledge of such scenario could be represented by extending standard computational logic with a basic time interval along with a shifting temporal operator, e.g. "after". Assuming then a hierarchy of scales, and using a Prolog-like notation the models of this scenario could look like, for example, as follows

where $t(1,1,1)$ stands for *first day of the first month of the first year*; $i(X...Y,S)$ represents a cyclical interval from X to Y at scale S; A @ T means A *is true throughout* T; $p(1,C)$ stands for *period of 1 unit at scale* C; ***altitude*** represents the vertical position of the bug, and $S...[T]$ represents a linear interval open on the right, i.e. T is not included; C_1 represent the

constraints on t_1's reaction to the attack of b_1; C_b represent the behavior of b_1 according to the height of t_1; $included(T, I)$ is true if T is a time included in interval I.

$season(raining)$ @ $i(12...4, month)$.
$value(height, t_1, 5)$ @ $t(1,1,1)$.
$value(height, t_1, H_f)$ @ $p(1, week)$ $after$ $T_p \Longleftarrow$
  $value(height, t_1, H_1)$ @ T_p &
  $P_1 = [V \mid value(altitude, b_1, V_1)$ @ T and
    $included(T, T_p...\lfloor p(1, week)\ after\ T_p \rfloor)]$ &
  $P_2 = [V \mid value(height, t_2, V_2)$ @ T and
    $included(T, T_p...\lfloor p(1, week)\ after\ T_p \rfloor)]$ &
  $C_1(H_1, P_1, P_2, H_f)$.
$value(height, t_2, 7)$ @ $t(10,1,1)$.
$value(height, t_2, H_f)$ @ $p(1, week)$ $after$ $T_p \Longleftarrow$
  $value(height, t_1, H_1)$ @ T_p &
  $P_1 = [V \mid value(height, t_1, V_1)$ @ T and
    $included(T, T_p...\lfloor p(1, week)\ after\ T_p \rfloor)]$ &
  $P_2 = \{V \mid value(altitude, b_1, V_2)$ @ T and
    $included(T, T_p...\lfloor p(1, week) ter\ T_p \rfloor)]$ &
  $C_2(H_1, P_1, P_2, H_f)$.
$value(altitude, b_1, 4)$ @ $t(1,5,1)$.
$valuue(altitude, b_1, 0)$ @ $T \Longleftarrow season(raining)$ @ T.
$value(altitude, b_1, A_f)$ @ $p(1, day)$ $after$ $T_p \Longleftarrow$
  $\neg$ $season(raining)$ @ $\lfloor T_p...p(1, day)\ after\ T_p \rfloor$.
  $value(altitude, b_1, A_p)$ @ T_p &
  $value(height, t_1, H_1)$ @ T_p &
  $value(height, t_2, H_2)$ @ T_p &
  $C_b(A_p, H_1, H_2, A_f)$.

The notation above is quite similar to standard computational logic. The differences are the introduction of a temporal operator, the functions "..." (using infix notation) to represent linear time intervals and i for cyclical intervals. Along with the meaning of specific terms which have to be carefully defined within the Herbrand universe (e.g. $i(12...2, month)$), this rich notation brings three interesting problems. First, how to specify a hierarchy of time scales where cyclical intervals are objects of the language? Second, what is the meaning of these programs? Third, how should they be interpreted considering that standard unification would not match different functions despite their intuitive meaning says they are equivalent?

This work aims to answer these questions by showing that we can relax the restriction placed on the possible interpretations of the equality theory with a sophisticated unification. For this, cyclical and granular time theories are seen as subsets of the Herbrand universe according to a special interpretation. We can understand the solution we shall present as a kind of generalization of Clark's completion.

We developed a logic based framework endowed with these features lacking in traditional approaches as we shall see in Section 2.

The rest of this paper is organized as follows. In Section 2 we shall relate some work on temporal planning and reasoning based on graph, satisfiability and temporal logic approaches. In Section 3, we shall see a definition for **NatureTime** along with a denotational meaning of its expressions. In Section 4 we shall see a framework to allow us work with **Nature** time normal programs, where there is a clear separation between standard and temporal unification. In Section 5 we shall see a meta-interpreter for our language and we address its properties. Finally, in Section 6 we shall make an analysis of the expressive power of our approach and some conclusions.

2 Related Work

We all know that a theory of granularity, [Hobbs, 1985], is formed by a set of local theories and how they *articulate* (or relate) to each other. Most the approaches proposed to formalise the connection of local theories and how they relate to each other, received influences from interval calculus [Allen, 1983] and *union-of-convex*. The latter an extension of the former.

The graph representation proposed, known as Temporal (interval-based) Constraint Network [Dechter et al., 1991], brought many interesting problems of tractability. Many efforts have been made to improve the path consistency algorithm (PCA) for temporal reasoning, e.g. [Morris et al., 1993]. Others have put some strength on domain features to reduce time complexity, like [El-Kholy and Richards, 1996] that showed a planning temporal and resource reasoning system, *parc*PLAN based on a temporal network which involves only binary constraints.

The focus on the efficiency of algorithms seems to have blurred the interest for more expressive representations such as granularity. For example, in the case of *parc*PLAN, constraint entailment depends on a minimal network model, computed by a PCA, to perform a propagation on time points variables. Recently, [Liatsos and Richards, 1999] investigated the limitations on the scaleability of the planning algorith of *parc*PLAN.

[Kautz and Selman, 1992] proposed that reasoning about planning can be better performed by propositional satisfiability. Instead of searching the exponentially large space of truth assignments for a model, this approach use logical representation that has good com-

putational properties. In [Kautz and Selman, 1999] an experiment on the unification of both, SAT-based and Graph-based (used in most time interval) approach.

[Ciapessoni et al., 1993] used a Topological Logic (TL)[1], to propose a many-sorted first order predicate calculus. This work extends TL with operators to identify the domain or level of granularity at which formulae should be considered, and also to constrain formulae to different domains.

The gains from this "wedding" are the use of temporal operators and a metric on time to deal with time granularity. The hierarchy of time is a linear structure called the universe of domains, where a *granularity ordering* relationship is imposed over this universe. This is similar to the local theories, suggested by Hobbs (*op. cit.*). This was shown to useful in the context of planning systems [Badaloni and Berati, 1994], in order to achieve plan actions at different scales of time and reduce the computational complexity of such systems. But, now we had the lack of temporal elements for cyclical events.

[Liu and Orgun, 1996] proposed Chronolog(MC) to deal with multiple granularity of time. This is based on a temporal logic with clocks in which we can associate a local clock with each predicate symbol. Every program is composed of a clock definition, a clock assignment and a program body. Granularity of time is more flexibly represented because there is no commitment that a grain of time is a refinement of another one. This can be problematic in some application domains where we have interacting processes working at different clocks.

This paper presents a framework for dealing with time in temporal logic programs developed in [Mota, 1998] and improves previous development of **NatureTime** logic [Mota et al., 1996] with a better treatment for temporal unification. This is a quantified temporal logic in which quantifiers ranges over moments and intervals, where special terms of the Herbrand universe stand for temporal relations. For this we introduce an environment to deal with temporal substitutions and a special temporal combination. The correctness of the meta-interpreter is addressed and we outline the suitability of this approach to deal with interacting agents non-aligned in time and a different granularities.

[1]Topological Logic extends standard propositional logic with a parameter operator P_α to mean that "proposition p is realised at the position α".

3 NatureTime Definition

3.1 Vocabulary

We extend the usual classical set of symbols for variables, say $\mathcal{L}_v$, with a countably infinite set of temporal variables $\mathcal{L}_{tv}$; the set of constants, say $\mathcal{L}_c$, with a finite set $\mathcal{L}_{tc}$ of temporal constants defined as $\{flow\_time, infinite, smallest\} \cup \mathcal{T}_C$, where $\mathcal{T}_C$ is a set of names of temporal classes; the set $\mathcal{L}_f$ of functions, with a finite set $\mathcal{L}_{tf}$ of temporal function symbols where an important subset of it is $\{.../2, p/2, i/2, t/k, after/2, plus/2\}$, the last two are written using infix notation; the set $\mathcal{L}_p$ of predicate symbols with the special predicates $mod\_temp\_class/3$, $future/3$.

The logical connectives for negation[2], conjunction, disjunction, and implication are represented here by $\neg$, $\&$, $\vee$, and $\Longleftarrow$, respectively, where truth value for true is represented by $\top$, and false by $\bot$; the temporal symbol @ (which connect terms to their temporal labels); finally the temporal interval demarcator $\lfloor\ \rfloor$.

3.2 Classes of Expressions

The definitions for logical term and atomic formula (AF) are the usual classical notions. The first kind of special expression we need is the definition for modular temporal classes. This is formed by using the special predicate $mod\_temp\_class/3$ as follows, where $\mathbf{Z}_m$ represents a modular set with m elements from the integers starting from 1 up to m.

Definition 1 (Modular Temporal Class) *Let c_i and c_j be constants. If we associate temporal units to c_i and c_j so that each unit of c_i contains m units of c_j (or a modular set $\mathbf{Z}_m$), then we can say that c_i is a* modular temporal class *(MTC) defined by a modular set of c_j, written $mod\_temp\_class(c_i, c_j, m)$.*

An important finite sequence of sentences defining MTCs have the form as follows, where $x...y$, $x, y \in \mathbf{Z}$ is a notation to denote a range between these numbers.

$mod\_temp\_class(flow\_time, c_n, infinite)$, $c_n \in \mathcal{T}_C$
$mod\_temp\_class(c_i, c_{i-1}, m_v)$, $1 < i \leq n$, $c_i, c_{i-1} \in \mathcal{T}_C$, and $m_v \in \mathbf{Z}^+$.
$mod\_temp\_class(c_1, smallest, 1)$, $c_1 \in \mathcal{T}_C$.

If c_{i-1} defines c_i with modular value m_v, then we say c_i is an *immediate descendant* of c_j. Any valid sequence of sentences defining MTCs plus the tempo-

[2]Negation is allowed by failure under the closed world assumption because this language is not for programming open agent-based systems yet.

ral unification (Section 4.4), is called a Modular Theory of Time (MoTT). In a MoTT there will always be a unique sequence of MTC definitions from (the lowest) c_1 to (the highest level) c_k called the *Main Time Hierarchy* (MTH). But we can augment a MoTT with other classes. For instance, suppose we choose $\mathcal{T}_C = \{day, month, year\}$, then we may assert.

$mod\_temp\_class(flow\_time, year, infinite)$.
$mod\_temp\_class(year, month, 12)$.
$mod\_temp\_class(month, day, 30)$.
$mod\_temp\_class(day, smallest, 1)$.

These relations define a MoTT where the flow of time, represented by *flow_time*, is a linear and infinite (*infinite*) sequence of *years*, *year* is a MTC of 12 *months*, *month* is a MTC of 30 *days*, *day* is the smallest (*smallest*) time interval. Note that months are regarded as regular MTC. Real Calendars with non-regular MTC can also be defined as shown in [Mota et al., 1996]. In what follows, some valid classes of expressions are defined and collection intervals are not treated in this work.

temporal term (TT) is a temporal variable $s \in \mathcal{L}_{tv}$, or a temporal constant $c_t \in \mathcal{T}_C$, or a

period, is recursively defined as a 1) *single period* $p(s, m)$ where $s \in \mathbb{Z}^+$ and $m \in \mathcal{T}_C$; or 2) a *composite period* P *plus* P', where P is *single period*, and P' is a *period* term. Let $\mathcal{P}$ be the set of elements of this form.

smallest temporal entity (STE) $t(x_1, \ldots, x_k)$, where for n as the number of elements in $\mathcal{T}_C$, then $k \leq n$, each $x_i \in \mathbb{Z}^+$, and each i correspond to exactly one element of $\mathcal{T}_C$. Let $\mathcal{T}_M$ be the set of elements of this form (moments of time).

linear interval $s_1 ... s_2$, where s_1 and s_2 are STEs. Let $\mathcal{I}_L$ be the set of elements of this form.

cyclical interval $i(y_1 ... y_2, c)$, where $c \in \mathcal{T}_C$, and either $y_1, y_2 \in \mathbf{Z}_m$; or $y_1 = \alpha(x_1)$ and $y_2 = \alpha(x_2)$, where $\alpha \in \mathcal{T}_C$ and $x_1, x_2 \in \mathbf{Z}_m$ such that $mod\_temp\_class(c, \alpha, m)$ holds. Let $\mathcal{I}_o$ be the set of elements of this form.

Examples of TTs are $p(1, day)$, $p(5, day)$ *plus* $p(1, week)$, $t(17, 7, 1994)$, $i(9...2, month)$ and $i(day(5)...day(1), week)$.

pure temporal expression (PTE) $\mathcal{P}_{TE} = \mathcal{T}_M \cup \mathcal{I}_L \cup \mathcal{I}_o \cup \{t \mid t$ is in the form p *after* t', where $p \in \mathcal{P}$, and and $t' \in \mathcal{P}_{TE}\}$. An example of a PTE is $p(24, year)$ *after* $t(17, 7, 1970)$.

non-explicit temporal moment is $\lfloor T \rfloor$ if either $T \in \mathcal{T}_M$ or it is in the form p *after* T' and $T' \in \mathcal{T}_M$. This is useful to delimit intervals where the right ending point is not included but the immediate moment before it is. For example, $attack(bug\_pest, tree)$ @ $T ... \lfloor p(1, week)$ *after* $T \rfloor$.

atomic temporal formula (ATF) is either an AF or an AF annotated with a PTE by using the temporal operator "@", i.e. if A is an AF and T is a PTE, then A @ T is an ATF.

body is in one of the forms A & B, $A \vee B$, C or $\neg C$, where A and B are bodies and C an ATF.

well formed temporal formula (WFTF) is either an ATF A with body equal to $\top$, or $A \Longleftarrow B$ where A is an ATF and B is a body. A is called the *head*. A WFTF which is composed only by AFs is a Prolog clause with no temporal contents. An example of a WFTF is

$$attack(bug\_pest, tree) \text{ @ } p(1, month) \text{ after } T$$
$$\Longleftarrow$$
$$eating(bug\_pest, leaves(tree)) \text{ @ } T \;\&$$
$$attack(p_j, bug\_pest) \text{ @ } T \;\&$$
$$resisted(bug\_pest, p_j) \text{ @ } T ... \lfloor p(1, month) \text{ after } T \rfloor.$$

This means *the bug_pest attacks tree 1 month after T if it was eating the leaves of the tree at T and it was attacked by pesticide p_j at T and it resisted the effects of p_j since T until the moment before the current attack.*

We shall see next an intuitive semantics of **NatureTime**. We shall not see a formal presentation for lack of space. More of this can be found in [Mota, 1998].

3.3 The Meaning of NatureTime Expressions

We interpret every non-temporal formula (or program) in the same way as the usual definite logic programs [Lloyd, 1993]. For practical reasons, this work assumes points as primitive temporal entities. We assume assume that points range over $\mathbf{Z}$, but other choices are possible (e.g. points as states of the world). We write $2^{\mathbf{Z}}$ as subsets of $\mathbf{Z}$. A point is represented as a set with only one element. An interval is a set of points with the first and last elements as the ending points of the interval.

The formal semantics of **NatureTime** maps temporal terms to sets of moment of time. Because cyclical intervals are represented at a given level of granularity they occur, then we map them to all linear intervals they represent at that level. Formally we have as follows.

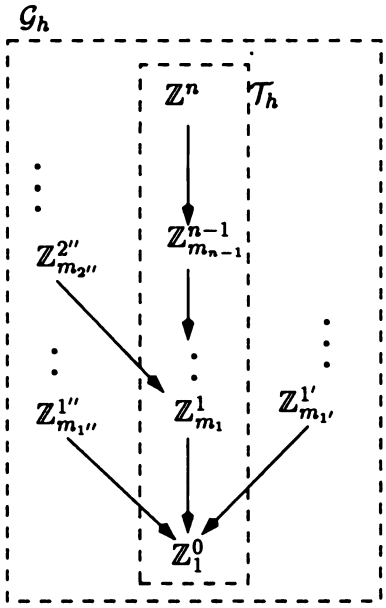


Figure 1: Induced graph of time granularity. The MTCs inside the innermost dashed box compose $\mathcal{T}_h$.

Definition 2 (Linear Collapsi Function) *Let $\mathbf{Z}_{m_1}, \mathbf{Z}_{m_2}, \ldots, \mathbf{Z}_{m_{n-1}}$ be modular sets, $\mathbf{Z}$ the set of positive integers. Then the* Linear Collapsi Function *μ_o is a mapping $\mathbf{Z}_{m_1} \times \mathbf{Z}_{m_2} \times \ldots \mathbf{Z}_{m_{n-1}} \times \mathbf{Z}$ to $\mathbf{Z}$. If $x_1 \in \mathbf{Z}^1_{m_1}, \ldots, x_n \in \mathbf{Z}$, then*

$$\mu_o(x_1, \ldots, x_n) = x_n \times m_{n-1} \ldots \times m_2 \times m_1 + x_{n-1} \times m_{n-2} \times \ldots m_2 \times m_1 \ldots + x_2 \times m_1 + x_1,$$

We relate the set of predicates defining modular temporal classes (presented in Section 3.2) to a tree $\mathcal{G}_h$, where nodes are modular sets and we have a partial ordering relation over them. By adding to this a distinguished leaf node with label $\mathbf{Z}$ we end up with a structure called *Linear-Cyclic Hierarchy* (LCH) of time, depicted in Figure 1. Thus, augmented $\mathcal{G}_h$ has a subset $\mathcal{T}_h$ which defines a unique directed path from the highest set of integers to the root.

A **NatureTime** temporal model $\mathcal{M}$ consists of a set $\mathcal{D}$ of non temporal individuals, a mapping ψ_c from $\mathcal{L}_c$ to $\mathcal{D}$, a LCH, a mapping τ_c from $\mathcal{L}_{tc}$ to $\mathcal{V}$, and a mapping $\tau_p : \mathcal{L}_p \times \mathcal{D}^n \times \mathcal{E} \rightarrow \{\top, \bot\}$.

A **NatureTime**-Interpretation is a triple $\mathcal{H}_{nt} = \langle \mathcal{M}, g, h \rangle$ where $\mathcal{M}$ is a temporal model and g is a function which assigns values to the variables of **NatureTime** such that g maps the set $\mathcal{L}_v$ of non-temporal variables to $\mathcal{D}$ and h maps the set $\mathcal{L}_{tv}$ of temporal variables to $2^{\mathbf{Z}}$.

The denotation of a well-formed expression α relative to a **NatureTime**-Interpretation $\mathcal{H}_{nt}$, written as $[\![\alpha]\!]^{\mathcal{H}_{nt}}$, is a mapping from $\mathcal{L}_v \cup \mathcal{L}_{tv}$ to $\mathcal{D} \cup 2^{\mathcal{E}}$ is defined as follows, where $|S|$ is the number of elements of set S.

Classical terms

- if $v \in \mathcal{L}_v$, then $[\![v]\!]^{\mathcal{H}_{nt}} = g(v)$
- if $c \in \mathcal{L}_c$, then $[\![c]\!]^{\mathcal{H}_{nt}} = \psi_c(c)$
- if $f \in \mathcal{L}_f$ and n is the arity of f then $[\![f(x_1, \ldots, x_n)]\!]^{\mathcal{H}_{nt}} = \psi_f(f)([\![x_1]\!]^{\mathcal{H}_{nt}}, \ldots, [\![x_n]\!]^{\mathcal{H}_{nt}})$

For temporal terms defined for this work, where $f \in \mathcal{L}_{tf}$, we have:

- if $v \in \mathcal{L}_{tv}$ then $[\![v]\!]^{\mathcal{H}_{nt}} = h(v)$
- if $c \in \mathcal{L}_{tc}$, then $[\![c]\!]^{\mathcal{H}_{nt}} = \tau_c(c)$
- if f has the form $t(x_1, \ldots, x_n)$, then $[\![t(x_1, \ldots, x_n)]\!]^{\mathcal{H}_{nt}} = \{\mu_o(x_1, \ldots, x_n)\}$
- if f has the form $t(x_1, \ldots, x_n)...t(y_1, \ldots, y_n)$, then $[\![t(x_1, \ldots, x_n)...t(y_1, \ldots, y_n)\ ]\!]^{\mathcal{H}_{nt}}$ is $\{z \in \mathbf{Z} \mid \mu_o(x_1, \ldots, x_n) \leq z \leq \mu_o(y_1, \ldots, y_n)\}$.
- if f has the form $i(x...y, c_j)$ where $x, y \in \mathbf{Z}_{m_j}$ of $c_j \in \mathcal{L}_{tc}$ and $[\![c_j]\!]^{\mathcal{H}_{nt}} \in \mathcal{T}_h$, then $[\![i(x...y, c_j)]\!]^{\mathcal{H}_{nt}}$ is $\bigcup\{I \mid$ where I is a linear interval from x to y at level $c_j\}$.
- if f has the form $p(s, c_i)$, $s \in \mathbf{Z}^+, c_i \in \mathcal{L}_{tc}$, $\mathbf{Z}_{m_i}, \mathbf{Z}_{m_{i-1}}, \ldots, \mathbf{Z}_1$ is path from the level of c_i until the root of $\mathcal{G}_h$, $I \in \mathcal{I}_L$ and it is ground, then $[\![p(s, c)]\!]^{\mathcal{H}_{nt}} = |[\![I]\!]^{\mathcal{H}_{nt}}|$ such that $|[\![I]\!]^{\mathcal{H}_{nt}}| = s \times m_i \times m_{i-1} \times \ldots \times 1$.
- if f has the form $p\ after\ r$ then

  $[\![p\ after\ r]\!]^{\mathcal{H}_{nt}} = [\![r]\!]^{\mathcal{H}_{nt}}\ \omega_{\oplus}\ [\![p]\!]^{\mathcal{H}_{nt}}$, where $\omega_{\oplus}$ is the up-wave modular sum. Basically, this operation propagates the "excess" of a sum at one level to the next above.

The denotation of classical and temporal formulae shall be:

- if $\alpha \in \mathcal{L}_p$ and n is the arity of α, then $[\![\alpha(x_1, \ldots, x_n)]\!]^{\mathcal{H}_{nt}} = \top$ iff $\tau_p(\alpha, \langle [\![x_1]\!]^{\mathcal{H}_{nt}}, \ldots, [\![x_n]\!]^{\mathcal{H}_{nt}} \rangle, t) = \top$ for all $t \in \mathcal{E}$.

This means that this AF is time-independent, and so does not need to be annotated with a PTE.

- if $\alpha(x_1,\ldots,x_n)$ @ T is an ATF and T a PTE, then $[\alpha(x_1,\ldots,x_n)\ @\ T]^{\mathcal{H}_{nt}} = \top$ iff $\forall t \in [T]^{\mathcal{H}_{nt}}$ $\tau_p(\alpha,[x_1]^{\mathcal{H}_{nt}},\ldots,[x_n]^{\mathcal{H}_{nt}}),t) = \top$ In other words, every temporal formulae A @ T means *A is true if A is true through the set of points to which T is mapped.*
- if A and B are ATFs, then $[A\&B]^{\mathcal{H}_{nt}} = \top$ iff
  $[A]^{\mathcal{H}_{nt}} = \top$ and $[B]^{\mathcal{H}_{nt}} = \top$.
  $[A \vee B]^{\mathcal{H}_{nt}} = \top$ iff
  $[A]^{\mathcal{H}_{nt}} = \top$ or $[B]^{\mathcal{H}_{nt}} = \top$.

Definition 3 (Logical Consequence) *A formulae B follows from a formula A iff every temporal model $\mathcal{M}$ and variable assignments g and h where A is true also makes B true.*

4 Temporal Substitutional Framework and Unification

Temporal unification handles labels and terms related to the temporal universe in a special way. From the point of view of classical logic, temporal labels and terms are quantified (not necessarily ground). In this case, the deductive system operates on constants and function terms of the Herbrand universe. Instead of having to test for equality between such expressions we use a test for unifiability between them. A similar idea was proposed in [Frisch, 1991], where a quantified term is interpreted as a schema for the set of its ground instances.

Sort theory provides mechanisms for making general claims about individuals in a certain domain class. Here, the elements of temporal domain are defined by special functions and constants. Thus we can make a general claim about temporal individuals in a restricted subset of the Herbrand universe to index possible states of the world. We shall see now a special environment to deal with it.

4.1 Temporal Variable and Substitutional Framework

In order to perform temporal reasoning by *semantic unification* of PTEs, a wftf is separated from its temporal terms so that temporal unification can be handled apart from standard unification. Every PTE which annotates an ATF is replaced by a temporal variable (*t-variable* for short). Then, the temporal term is treated as a binding for that variable. As a consequence substitution has a different treatment when involving *t-variables*. There is little difference between the usual substitution, as presented in [Lloyd, 1993], and temporal substitution as the following definition will show.

Definition 4 (TSF) *A* Temporal Substitutional Framework (TSF) *is a finite set of the form $\{tv_1/tb_1,\ldots,tv_n/tb_n\}$, where each $tv_i \in \mathcal{L}_{tv}$ is a* t-variable, *each $tb_i \in \mathcal{P}_{TE}$ is a term, called* t-substitution, *different from tv_i and $tv_1,\ldots,tv_n$ are distinct.*

For clarity sake, *t-bound* is used for a *t-variable* which is bound to a PTE. A *t-variable* is *t-free* if its corresponding temporal binding is non instantiated Prolog variable, and it is *t-bound* or temporally instantiated if its binding is a PTE. We should interpret such bindings as *schema* for temporal information. Suppose we have the formula A @ T_1 and *t-substitution* $\{(T_1, i(1...4, month))\}$. A classical *substitution instance* for this would be A @ $i(1...4, month)$. If we want to resolve this formula with A @ T_2 & B @ T_2 with *t-substitution* $\{(T_2, i(3...6, month))\}$, then the resolvent formula should be B @ T_1 with *t-substitution* $\{(T_1, i(3...4, month))\}$.

To obtain this we do not apply standard substitution and also the application of *t-substitution*, in the usual sense, is delayed until the very end of the deduction. During deduction, the set of correct *t-substitutions* is composed by manipulating the bindings of *t-variables* according to the underlying MoTT. Thus we standardize temporal formula so that the substitutions may be consistently generated[3]. For this the restrictions over the range of a *t-variable* and PTEs are now defined.

4.2 Restrictions on T-variables and Temporal Normal Form

The restrictions operate on *t-variables* wherever they occur in a temporal clause and within temporal entities because of their structured nature. Thus, when dealing with sentences of the form A @ T, while A can be any classical formula representing an event, a process, an action, etc., T ranges over $\mathcal{P}_{TE}$. If $T \in \mathcal{T}_M \cup \mathcal{I}_L \cup \mathcal{I}_o$, then the restrictions of the components for each case are those presented in Section 3.2. However, if T is a term the form P *after* T, then P and T cannot be both *t-variables* at the same time. If P is a non-instantiated variable, then T must be *t-bound*. But if

[3]Such a mechanism improves previous versions of **NatureTime.** as we mentioned in Section 2.

T is *t-free*, then P must be instantiated to a ground term.

The last two restrictions are useful to obtain which period of time is related to T and the TE represented by P *after* T, and to know which TE in the past (in this case T) is related to a TE in the future (in this case P *after* T) and the period between them (i.e. P), respectively. Table 1 summarizes the allowed forms of PTEs (regarded in this work) and their corresponding representation as temporal variables in a TSF (TSF-term for short). In what follows CPTE stands for canonical PTE; X and Y are variables; expressions in the form $i(Z_1...Z_2, C)$, $t(X_1, \ldots, X_n)$; $S...T$ where $S, T \in \mathcal{T}_M$, are CPTEs; a *t-substitution* tv/t_b is represented by (t_v, t_b).

Table 1: Allowed Forms of Pure Temporal Expressions/Corresponding Normal Forms.

PTE	TSF-term	Restriction
X	$\{(X,Y)\}$	$var(X)$,$var(Y)$
$i(Z_1...Z_2, C)$	$\{(X,$ $i(Z_1...Z_2, C))\}$	$Z_1, Z_2 \in \mathbf{Z}^+$, $C \in \mathcal{T}_C$ or $var(Z_1)$
$t(X_1, \ldots, X_n)$	$\{(X,$ $t(X_1, \ldots, X_n))\}$	$X_i \in \mathbf{Z}_{mi}$, $C_i \in \mathcal{T}_C$ or $var(X_i)$ for $i = 1, \ldots, n$
$S...T$	$\{(X, S...T)\}$	$S, T \in \mathcal{T}_M$ or $var(S)$ and $var(T)$
P *after* T	$\{(X, P$ *after* $T)\}$ $\{(X, P$ *after* $T)$, $(T,Y)\}$	T is a CPTE or *ground*(P) and $var(T)$

Now we can define the standard form we need for temporal formula. For clarity sake we shall use union of sets when appropriate.

Definition 5 (Temporal Normal Form) *Let A be a WFTF and $T_1, ..., T_n$ PTEs occurring in A. $A' :: \theta$ is the* temporal normal form (TNF) *of A, where A' is the result of replacing every ATF A_i @ T_i in A with A_i @ X_i where X_i is a new variable, $\theta_s = \{(X_1, T_1), \ldots, (X_n, T_n)\}$ is the t-substitution.*

The computation which transforms a temporal formula into its TNF is given by the Algorith 1

In what follows we have the transformation of the WFTF of Section 3.2 by using **Algorithm** 1.

By applying rule 6 to the head of the clause results in the following expression.

Algorithm 1 Rewrite a Temporal Formula in to a TNF using the following rule set:

Require: A is a WFTF.
Ensure: $A' :: \theta$ is the TNF of A.
Rule 1: A is a classical AF
$A \rightarrow A :: \varnothing$, or $A \rightarrow A \Longleftarrow \top :: \varnothing$, in the case A is also an assertion.
Rule 2: A is A @ $T_1...T_2$ and $T_1, T_2 \in \mathcal{T}_M \cup \mathcal{I}_L \cup \mathcal{I}_o \cup \mathcal{P}_{TE}$
A @ $T_1 \rightarrow A$ @ $X_1 :: \theta_1$
A @ $T_2 \rightarrow A$ @ $X_2 :: \theta_2$
A @ $T_1...T_2 \rightarrow A$ @ $T :: \{(T, X_1...X_2)\} \cup \theta_1 \cup \theta_2$
Rule 3: A is A @ T and T is a standard variable
A @ $T \rightarrow A$ @ $T :: \{(T,V)\}$, where V is a new variable for the binding of T.
Rule 4: A is A @ T_e and T_e is a canonical PTE
A @ $T_e \rightarrow A$ @ $T :: \{(T, T_e)\}$, T is a new temporal variable with binding T_e.
Rule 5: A is A @ P *after* T_e and either P is a ground period and T_e a variable or
T_e is a canonical PTE or $T_e \in \mathcal{T}_M \cup \mathcal{I}_L \cup \mathcal{I}_o$
A @ P *after* $T_e \rightarrow A$ @ $T :: \{(T, P$ *after* $T_e)\}$
Rule 6: A is A @ P_1 *after* T and $T \in \mathcal{P}_{TE}$
A @ $T \rightarrow A$ @ $T_2 :: \theta_2$
A @ P_1 *after* $T \rightarrow A$ @ $T_1 :: \{(T_1, P_1$ *after* $T_2)\} \cup \theta_2$
if $T \in \mathcal{P}_{TE}$
Rule 7: A is A @ $T_1 :: \theta_1$ & B @ $T_2 :: \theta_2$ and V is the binding of T_1 and T_2
A @ $T_1 :: \theta_1$ & B @ $T_2 :: \theta_2 \rightarrow A$ @ T_2 & B @ $T_2 :: \theta_1 \setminus \{(T_1, V)\} \cup \theta_2$
Rule 8: A is $A_1 :: \theta_1$ & $A_2 :: \theta_2$
$A_1 :: \theta_1$ & $A_2 :: \theta_2 \rightarrow A_1$ & $A_2 :: \theta_1 \cup \theta_2$.
Rule 9: A is $A_1 :: \theta_1 \vee A_2 :: \theta_2$
$A_1 :: \theta_1 \vee A_2 :: \theta_2 \rightarrow A_1 \vee A_2 :: \theta_1 \cup \theta_2$.
Rule 10: $A :: \theta \Longleftarrow B :: \delta$
$A :: \theta \Longleftarrow B :: \delta \rightarrow A \Longleftarrow B :: \theta \cup \delta$.

$$
\begin{aligned}
&attack(b_1, t_1) \text{ @ } T_1 \\
&\quad :: \{(T1, p(1, month) \text{ after } T)\} \cup \{(T, V)\} \\
&\Longleftarrow \\
&eating(b_1, leaves(tree)) \text{ @ } T \ \& \\
&attack(byern, b_1) \text{ @ } T \ \& \\
&resisted(b_1, byern) \text{ @ } T...p(1, month) \text{ after } T.
\end{aligned}
$$

By applying rule 3 to the first two ATFs and rule 2 to last ATF of the body of the clause:

$$
\begin{aligned}
&attack(b_1, tree) \text{ @ } T_1 \\
&\quad :: \{(T_1, p(1, month) \text{ after } T), (T, V)\} \\
&\Longleftarrow \\
&eating(b_1, leaves(tree)) \text{ @ } T_2 :: \{(T_2, T)\} \ \& \\
&attack(byern, b_1) \text{ @ } T_3 :: \{(T_3, T)\} \ \& \\
&resisted(b_1, byern) \text{ @ } T_4 \\
&:: \{(T_4, T_5...T_6), (T_6, p(1, month) \text{ after } T_5), (T_5, T)\}.
\end{aligned}
$$

If we apply rule 8 twice in the body of the clause and then apply step 7 also twice we have.

$$attack(b_1, tree) \ @ \ T_1$$
$$:: \{(T_1, p(1, month) \ after \ T), (T, V)\}$$
$$\Longleftarrow$$
$$eating(b_1, leaves(tree)) \ @ \ T_5 \ \&$$
$$attack(byern, b_1) \ @ \ T_5 \ \&$$
$$resisted(b_1, byern) \ @ \ T_4$$
$$:: \{(T_4, T_5 ... T_6), (T_6, p(1, month) \ after \ T_5), (T_5, T)\}.$$

Note that some names of *t-variables* disappeared because they referred to the same binding due to step 7.

Step 4 - By applying rule 10 to whole clause we end up with the following WFTNF.

$$attack(b_1, tree) \ @ \ T_1$$
$$\Longleftarrow$$
$$eating(b_1, leaves(tree)) \ @ \ T \ \&$$
$$attack(byern, b_1) \ @ \ T \ \&$$
$$resisted(b_1, p_j) \ @ \ T_4$$
$$:: \{(T_4, T ... T_1), (T_1, p(1, month) \ after \ T), (T, V)\}.$$

Note that the binding of T is a variable V which denotes that T is *t-free* because T is not a ground term in the original WFTF. A temporal variable only appears in the binding of another temporal variable in the case of a non CPTE, e.g. $\{(X, p(1, month) \ after \ T), (T, V)\}$. These expressions are reduced to their minimal representation, i.e. the simplest structured term (or PTE) that it can be reduced as soon as the temporal variable T is instantiated to some non canonical form. Such a minimal form is a member of $\mathcal{I}_o \cup \mathcal{I}_M \cup \mathcal{I}_L$ according to the restrictions seen above.

Definition 6 (Temporal Normal Logic Program) *A* Temporal Normal Logic Program *(TNLP) $\mathcal{P}$ is a finite set of temporal clauses in their TNF, i.e. $\{C_1 :: \theta_1, \ldots, C_n :: \theta_n\}$. $\mathcal{P}^\beta$ is written as a short notation for a TNLP, where β is the family $\{\theta_1, \ldots, \theta_n\}$.*

We may compare this to Clark's Completion [Lloyd, 1993]. We just need to relax the restriction upon the possible interpretations of "=" with a theory by using a meta-language where negation is also interpreted as failure. This is enough to guarantee that the object language has the same interpretation. This is the case of normal **NatureTime** programs, and this theory is a MoTT which manipulates specific objects of the Herbrand universe (U_P). Here, we force the temporal combination (we shall see in Section 4.4) to be our identity relation on U_P.

4.3 Least Form of a PTE

A temporal variable can be *t-bound* to any temporal expression, for instance $P \ after \ T$, and there may be *t-variables* within such expressions which are bound to other terms. Hence we must get their *reduced form* before temporal unification. Cyclical intervals do not necessarily have a least form, but rather a minimal set of least forms. Because of such restrictions, the algorithm computes only those canonical forms established. This means that the termination of a least form computation only depends on the length of expressions of the form $P \ after \ T$, which is finite and so it terminates.

The least form of a PTE is the minimal representation of it, i.e. the simplest structured term that it can be reduced. Such a minimal form is called canonical temporal entity (CTE) and it is a member of $\mathcal{I}_o \cup \mathcal{I}_M \cup \mathcal{I}_L$ according to the restrictions of Table 1 (rows 1,2 and 3). Note that cyclical or collection intervals do not necessarily have a least form, but rather a minimal set of least forms. Because of such restrictions, the algorithm computes only those canonical forms established in this table. This means that the termination of a least form computation only depends on the length of expressions of the form $P \ after \ P$ (4th row of Table 1), which is finite. If the input is not a well formed temporal expression then the computation returns false.

For instance, if $\theta = \{(T, p(1, month) \ after \ X), (X, t(1, 1, 1))\}$, then the least form of $p(1, month) \ after \ X$ is $t(1, 2, 1)$. If X was bound to a variable the expression would not change.

The correctness of **Algorithm** 2 is based on two things. First is the constraints of Table 1 dealt by each case of the algorithm. Second, is the reduction of expression of the form $P \ after \ T$, $reduce(P \ after \ T, V)$. The reduction algorithm relies on the computation which relates a moment of time, another moment in the future and the period of time between them.

Such a computation $(future(T_p, P, T_f))$, always terminate because the up-wave modular sum (or subtraction if it is the case) either returns false or the *waving* effect of the sum (or subtraction) stops in a parameter of $t(x_1, \ldots, x_n)$ which corresponds to the highest (or lowest) level of time granularity. As the number of time scale is finite, if the constraints are obeyed, then the reduction finds a reduced form of the expression otherwise it returns false. Either $P \ after \ T$ is the least form where restriction 4.b is satisfied, or $P \ after \ T$ is reduced to a term of the form $t(x_1, \ldots, x_n)$ because 4.a is satisfied.

Algorithm 2 Computation of a least Form of a PTE

Require: θ is a TSF and T is a temporal expression.
Ensure: A least form of T is V_f in relation to θ, written $least\_form(\theta, T, V_f)$.
It returns false otherwise.
Case 1: $least\_form(\_, V, V)$
$var(V)$ or $canonical\_te(V)$.
Case 2: $least\_form(\_, P\ after\ T, V_f)$
if $ground(P)$ and $ground(T)$ **then**
$reduce(P\ after\ T, V_f)$
end if
Case 3: $least\_form(\theta, P\ after\ T, V_f)$
if $ground(P)$ and $var(T)$ **then**
$ground\_pte(\theta, T, GT)$
$reduce(P\ after\ GT, V_f)$
end if
Case 4: $least\_form(\theta, P\ after\ T, P\ after\ T_s)$
if $ground(T)$ and $var(P)$ **then**
$T_s = T$
else if $ground(P)$ and $var(T)$ and $\neg$ $ground\_pte(\theta, T, \_)$ **then**
T_s is $t\_bind(T, \theta)$
else if $ground(P)$ and $var(T)$ and $\neg$ $ground\_pte(\theta, T, \_)$ and
$t\_bind(T, \theta)$ is a non instantiated variable
then
$T_s = T$
end if

4.4 Temporal Combination

Suppose we have $A \Longleftarrow A_1 \wedge \ldots \wedge A_n :: \theta$, an instance of a temporal clause, and we want to solve A_i with an instance of another temporal clause $B \Longleftarrow C :: \sigma$. Assuming A_i is X_i @ T_i and B is X @ T. The temporal unification between T_i and T takes the underlying MoTT into account to perform a *semantic unification* between T_i and T, and apply its result to θ and σ. Since T_i and T are special terms of the Herbrand Universe (in the form of PTE as defined in Section 3.2), then terms which in Prolog unification would not unify, here may refer to the same temporal entities of a given MoTT(a), and so are equivalent

We can understand this special resolution step as similar to the clocked TiSLD-Resolution of Chronolog(CM) [Liu and Orgun, 1996]. The difference is that a "**NatureTime**-Resolution" needs a more elaborated unification algorithm between temporal terms because these ones are not simply natural numbers. The algorithm to unifying temporal terms has the following basis, where $temp\_unify/3$ performs a time matching or temporal reasoning algorithm (see [Mota et al., 1996, Mota, 1998] for details).

Definition 7 (Temporal Combination) *Let X and Y be* t-variables, *with bindings B_x and B_y, from two distinct* TSFs *θ and σ, respectively. A* Temporal Combination *between σ and θ by* temporally unifying *X and Y is a set $(\theta \backslash \{(X, B_x)\}) \cup (\sigma \backslash \{(Y, B_y)\}) \cup \{(Y, V))\}$, such that $temp\_unify(B_x, B_y, V)$ holds, i.e. V is the temporal unification between B_x and B_y.*

For example, suppose we have a TSF $\theta_1 = \{(T, i(9...6, month)), (Z, X)\}$ and another TSF $\theta_2 = \{(S, i(4...7, month))\}$. Then a temporal combination between θ_1 and θ_2 resulting from the temporal unification between the pair associated with T and S is $\{(T, i(4...6, month), (Z, X))\}$. Another possible temporal combination obtained from temporally unifying S and Z is $\{(T, i(9...6, month)), (Z, i(4...7, month))\}$. Note that a temporal combination does not necessarily find a unique temporal substitution. In the case of cyclical intervals their least form can be a (minimal) set, which is at most of size two.

For simplicity, the correctness of this algorithm is restricted only to ground temporal terms, i.e. cyclical intervals and linear intervals without variables. It is based on the correctness of the algorithm to reduce an expression to a minimal form, and of $temp\_unify/3$. Because of the restrictions imposed, then the algorithm either finds a temporal term or returns false.

Theorem 1 (Correctness) *Let θ, θ_v, θ_s be TSFs. Suppose $(T_v, V) \in \theta$, $(T_v, B_v) \in \theta_v$, $(T_s, B_s) \in \theta_s$, where V, B_v and B_s satisfy the restrictions shown on Section 4.2, and $\theta = (\theta_s \backslash \{(T_s, B_s)\}) \cup (\theta_v \backslash \{(T_v, B_v)\}) \cup \{(T_v, V))\}$. Then, the temporal combination between T_v and T_s wrt θ_v and θ_s is true and θ is computed iff $[V]^{\mathcal{H}_{nt}} \subseteq [B_s]^{\mathcal{H}_{nt}} \cap [B_v]^{\mathcal{H}_{nt}}$. Also,*

> *if either B_v or B_s are PTEs which involve only time moments (2nd and 4th row of* Table 1*), then V is a unit set, otherwise*
>
> *if B_v and B_s are PTEs which involve cyclical intervals, then V may be a set of two values.*

The proof of this theorem is not shown here, but we can have an intuition of how this can be done if we relate **NatureTime** temporal unification (or combination) with Chronolog(MC) temporal unification. In this approach temporal atoms are not annotated with temporal terms, but they are clocked and the unification takes clocks definition and clock assignment into account. These, as Liu and Orgun suggested, can be seen as procedures attached to Chronolog(MC) program body.

In **NatureTime**, this would be equal to splitting the temporal annotation of temporally sensitive predicates, and attaching the temporal reasoning mechanism in to the programs, also, as procedure calls. The basic difference is that **NatureTime** uses a more elaborated notation of temporal terms, and so another level of computation is necessary for unification. Furthermore, a clocked fixed-time atom in Chronolog(MC) is similar to an ATF in **NatureTime** where the temporal term is a moment of time. The notion of *answer substitution* used here is the same as presented in [Lloyd, 1993].

Definition 8 (Temporal Answer Substitution) *Let β be an answer substitution, $\mathcal{P}^{\beta}$ be a TNLP and $A :: \theta$ a temporal goal formula in its TNF. Then a* Temporal Answer Substitution *$\mathcal{P}^{\beta} \cup \{A :: \theta\}$ is a t-substitution for t-variables in θ obtained from the temporal combination between θ using the elements of β.*

Definition 9 (Correct t-substitution) *Let $\mathcal{P}^{\Gamma}$ be a temporal normal logic program, G be a goal $\Longleftarrow A_1 \& \ldots \& A_n :: \theta$, σ and δ be answer substitution and t-substitution for $\mathcal{P}^{\Gamma} \cup \{G\}$, respectively. We say σ and δ are a* correct substitution *and a* correct t-substitution *for $\mathcal{P}^{\Gamma} \cup \{G\}$ if $\forall(A_1 \& \ldots \& A_n)\sigma :: \delta$ is a logical consequence of $\mathcal{P}^{\Gamma}$.*

We shall see now a meta-interpreter which implements a proof procedure to find a correct *t-substitution* for a temporal goal.

5 A Meta-Interpreter for NatureTime Logic

The meta-interpreter is an extension of the one commonly used by logic programmers and presented by [Sterling and Shapiro, 1995] and elsewhere. The main extension is related to a special unification for PTEs. Because of this, the standard *solve*/1 predicate is changed to be a triple relation *solve*/3, where the first two arguments are for a temporal formulae in its TNF, i.e. the first is the temporal formula with temporal variables and the second is its corresponding TSF. The third is an answer substitution.

In what follows, $temp\_formula(X)$ is true if X is a wftf, $var(X)$ is true if X is a variable, and $clause(X \Longleftarrow Y :: U_n)$ is true if there is a clause with X as its head and Y as its body. If Y is $\top$ then X is an assertion, and for non temporal formula we assume that U_n is empty.

1 - $solve(\top, \theta, \theta)$.

2 - $solve(\neg X, \theta, \theta) \Longleftarrow \neg\ solve(X, \theta, \theta)$.

3 - $solve(X, \theta_s, \theta_f) \Longleftarrow$
$clause(X \Longleftarrow Y :: [\,]) \ \&\ solve(Y, \theta_s, \theta_f)$.

4 - $solve(X \ @\ T_s, \theta_s, \theta_f) \Longleftarrow$
$clause(X \ @\ T_p \Longleftarrow P :: \theta_p)\ \&$
$temp\_comb(T_s, \theta_s, T_p, \theta_p, \theta_n)\ \&\ solve(P, \theta_n, \theta_f)$.

5 - $solve(A \ \&\ B, \theta_s, \theta_f) \Longleftarrow$
$solve(A, \theta_s, \theta_n)\ \&\ solve(B, \theta_n, \theta_f)$.

6 - $solve(A \vee B, \theta_s, \theta_f) \Longleftarrow$
$solve(A, \theta_s, \theta_f) \vee solve(B, \theta_s, \theta_f))$.

The fourth clause can be seen as a **NatureTime**-Resolution which is similar to clocked TiSLD-Resolution because the resolution step is also subdivided into two steps: matching the predicate using standard unification and the temporal combination. The declarative interpretation for clause 4 is as follows.

4 - *X is true throughout T_s in θ_s with a set of temporal answer substitutions θ_f, if $X \ @\ T_p$ is the head of a temporal clause with body P and θ_p as its TSF, θ_n is a temporal combination between θ_s, considering t-variable T_s, and θ_p, considering* t-variable *T_p, and θ_f is the temporal substitution found by solving P with TSF θ_n.*

All other clauses have the usual interpretation for meta-interpreters. The correctness of this interpreter is restricted to canonical instances of temporal normal formulae. A similar idea to Chronolog(MC).

Theorem 2 (Correctness of Meta-Interpreter) *Let $\mathcal{P}^{\Gamma}$ be a temporal normal logic program, G be a goal $\Longleftarrow A_1 \& \ldots \& A_n :: \theta$, σ and δ be answer substitution and t-substitution for $\mathcal{P}^{\Gamma} \cup \{G\}$, respectively. Then,* **NatureTime** *meta-interpreter will find σ and δ as* correct substitution *and a* correct t-substitution *for $\mathcal{P}^{\Gamma} \cup \{G\}$ iff $(A_1 \& \ldots \& A_n)\sigma :: \delta$ is a member of all canonical instances of $\mathcal{P}^{\Gamma}$.*

The proof of this theorem, can be done by induction on the length of the derivation steps of the *solve*/3 procedure. This correctness is also restricted to the same limits of standard logic programs. Programs like

$die(t_1)\ @\ T \Longleftarrow \neg die(t_2)\ @\ T :: \{(T, V)\}$.
$die(t_2)\ @\ T \Longleftarrow \neg die(t_1)\ @\ T :: \{(T, V)\}$.

are not considered, even if the temporal terms are ground instances of a time moment term or a cyclical interval.

6 Discussions and Conclusions

The expressive power of a framework like **NatureTime** allows us to take advantage of the fact that a large class of ecological simulation models can be represented using a standard clause schema which we call a *simulation clause*[Mota, 1998]. Usually the specification of a simulation model to compute the value, V, of an attribute, Att, of a given agent, Ag, at a time, P after T is in the form

$$value(Att, Ag, V_i) \ @ \ T_i.$$
$$(value(Att, Ag, V) \ @ \ (P \ after \ T_p))$$
$$\Longleftarrow (value(Att, Ag, V_p) \ @ \ T_p \ \& \ \mathcal{R}(V_p, V)).$$

where the first clause states that the value of the attribute Att of Ag is V_i at the initial time T_i. In the second clause, P is a length of time of size 1 at some scale; $after$ is a displacement function from T_p to the time P units later; and $\mathcal{R}(V_p, V)$ represents a sequence of formulae which imposes constraints on the clause (including recursive subgoals which may be for values of the attributes of other agents, but not for Att of Ag) and calculate a value for V from that of V_p. So, the first expression says the value of the attribute is V at time P after T_p if (represented by $\Longleftarrow$) the conditions in the body hold.

Note that the constraint on the existence of only one recursive subgoal relating Att of Ag establishes the threshold of the class of models of ecosystems our approach is able to represent. Using such schema we can specify the behavior of interacting aligned and non-aligned agents working at different scales of time. The mechanisms of inference which are necessary to deal with such agents can be implemented by just extending the **NatureTime** meta-interpreter [Mota, 1998].

Such mechanisms must be able to find the value of an agent's attribute at a given specific time (which may be in between two consecutive time steps), and the changes of such an attribute during a certain period of time for aligned and non-aligned periods in relation to changes on the agent's attributes). The challenge is basically to conveniently guide the search to find precisely the values we want, and for the period requested. obtained bye extending predicate *solve*/3 to deal with temporal search aligned and non-aligned in time. Others approach would have to first solve the problem of representing granularity and cyclicity and then tackle these issues.

The temporal substitutional framework separates standard and temporal unification in order to make a clear and consistent treatment of temporal entities within the same clause specification. This meta-interpreter shows more clearly than in [Mota et al., 1996] where the standard and temporal unification are treated. This approach takes advantage of a special subset of the Herbrand universe to represent and build time granularity theories which is defined by means of modular sets. As a consequence cyclical aspect of time is already embodied within the model. We also gave an intuitive but comprehensive outline of the correctness of the temporal combination and of the meta-interpreter.

Of course that an interpreter for **NatureTime** should deal with computations of processes aligned and non-aligned in time. However, dealing with interacting agents is not on the scope of this paper, although the results presented makes the framework remarkably powerful to this task.

Acknowledgements

We are most thankful to David Robertson and Alan Smaill for endless discussions and comments which helped us improve a lot our initial ideas, and to the referees for invaluable questions.

References

[Allen, 1983] Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11).

[Badaloni and Berati, 1994] Badaloni, S. and Berati, M. (1994). Dealing with time granularity in a temporal planning system. In *First International Conference on Temporal Logic*, pages 101–116, Bonn, Germany. Springer-Verlag.

[Ciapessoni et al., 1993] Ciapessoni, E., Corsetti, E., Montanari, A., and Pietro, P. S. (1993). Embedding time granularity in a logical specification language for synchronous real-time systems. *Science of Computer Programming*, 20(1):141–171.

[Dechter et al., 1991] Dechter, R., Meiri, I., and Pearl, J. (1991). Temporal constraints networks. *Artificial Intelligence*, 49(1):61–95.

[El-Kholy and Richards, 1996] El-Kholy, A. and Richards, B. (1996). Temporal and resource reasoning in planning: the *parc*PLAN approach. In *12th European Conference on Artificial Intelligence*, pages 614–618.

[Frisch, 1991] Frisch, A. M. (1991). The substitutional framework for sorted deduction: fundamental results on hybrid reasoning. *Artificial Intelligence*, 1(49).

[Hobbs, 1985] Hobbs, J. R. (1985). Granularity. In *Proceedings of the 9th IJCAI*, pages 1–4.

[Kautz and Selman, 1992] Kautz, H. and Selman, B. (1992). Planning as satisfiability. In *10th European Conference on Artificial Intelligence.*

[Kautz and Selman, 1999] Kautz, H. and Selman, B. (1999). Unifying sat-based and graph-based planning. In *16th International Joint Conference on Artificial Intelligence.*

[Liatsos and Richards, 1999] Liatsos, V. and Richards, B. (1999). Scaleability in planning. In *5th European Conference on Planning.*

[Liu and Orgun, 1996] Liu, C. and Orgun, M. A. (1996). Dealing with multiple granularity of time in temporal logic programming. *Journal of Symbolic Computation*, 22(5–6).

[Lloyd, 1993] Lloyd, J. W. (1993). *Foundations of Logic Programming.* Springer Verlag.

[Moore, 1995] Moore, R. C. (1995). *Logic and Representation.* CSLI Publications.

[Morris et al., 1993] Morris, R. A., Shoaff, W. D., and Khatib, L. (1993). Path consistency in a network of non-convex intervals. In *Proceedings of the of the IJCAI*, pages 655–660.

[Mota, 1998] Mota, E. (1998). *Time Granularity in Simulation Models within a Multi-Agent System.* PhD thesis, Department of Artificial Intelligence/The University of Edinburgh. PhD Thesis.

[Mota et al., 1996] Mota, E., Robertson, D., and Smaill, A. (1996). NatureTime: Temporal granularity in simulation of ecosystems. *Journal of Symbolic Computation*, 22(5–6):665–698.

[Robertson et al., 1991] Robertson, D., Bundy, A., Muetzelfeldt, R., Haggith, M., and Uschold, M. (1991). *Eco-Logic Logic-Based Approaches to Ecological Modelling.* The MIT Press.

[Sterling and Shapiro, 1995] Sterling, L. and Shapiro, E. (1995). *The Art of Prolog.* The MIT Press.

A Model for Reasoning about Topologic Relations between cyclic intervals

Philippe Balbiani and Aomar Osmani
Laboratoire d'informatique de Paris-Nord
Avenue Jean-Baptiste Clément, F-93430 Villetaneuse.
E-mail: ao@lipn.univ-paris13.fr

Abstract

This paper presents an algebraic framework for reasoning about topological relations between intervals defined in circle: the c_intervalles. After the study of the fundamental operations of reverse, intersection and composition, we will propose a neighborhood structure (v_structure) between the relations of the algebra. This structure allows the extraction of subclasses relations who present good properties: the convex and the preconvex relations. Adapting the propagation techniques designed to solve the networks of constraints between intervals, it shows that the problem of proving the consistency of c_intervals networks is NP-complete. In order to extract a tractable subclass of the algebra, we have defined a bijective function between every arc of a circle and points situated inside the circle, it proves that the convex relations are determined by areas inside the circle which can be given by the spiral movement of segments aligned with its center. Finally, it proposes a set of arc networks, the nice networks, which consistency can be decided in polynomial time by means of the path-consistency algorithm.

1 Introduction

Among the formalisms for reasoning about time used in Artificial Intelligence, the formalism using the constraint satisfaction techniques (CSP) takes up a very significant place. Indeed, in much of applications such as planning, natural language, scheduling problems, temporal knowledge is expressed in the form of a collection of binary relations between intervals or points. In this case, the temporal reasoning uses the CSP algorithms (path-consistency, k-consistency) to maintain the consistency of the relations (satisfiability), to seek a solution or a set of solutions satisfying all the constraints or to generate new relations from those given. In his article published in 1983, Allen [1] isolated the basic relations which can exist between intervals and proposed a variant of the path-consistency algorithm to maintain the consistency and to seek for solutions of the problems expressed in the form of the temporal constraint networks. Since, a multitude of works are devoted to the search of subclasses tractable in polynomial time and to the improvement of the algorithm of resolution proposed by Allen. The most notable works are: the introduction of the pointisable algebra by Vilain and Kautz [12], the proposal of some efficient algorithms for the search for solutions by van Beek [11], the definition of a maximum tractable subclass by Nebel and Bürckert [9], the definition of the preconvex subclass by Ligozat [7]. Other works studied various extensions of the interval algebra. The extensions of Ladkin [5], Morris [8], Ligozat [6] and Balbiani and al. [2] are the most notable extensions.
Constraint satisfaction techniques are also considered in spatial reasoning. The model of the areas proposed by Cohn, Cui and Randell [4] is, in this side, the well-known formalism to reason qualitatively about space. The issue of the tractability of the model proposed by Randel, Cui and Cohn has been adressed by Renz and Nebel [10].

This paper presents a formalism for representing and reasoning about interval described in closed curves.
The objects of this algebra are called c_intervals and are represented by arcs on the circle (see figure 1). This logic of intervals can express the links between events that recur in a cyclic way and which can be represented by a periodic function or to express the relations between objects described in closed curves. This algebra can be mixed with classical interval formalism to reason about several geometrical objects.

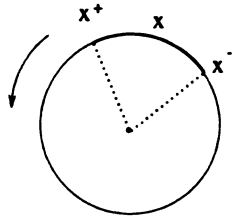


Figure 1: Example of c_interval $x = (x^-, x^+)$.

The figure 2 gives an example of spherical rectangle represented by cyclical intervals. For instance, this representation makes it possible to reason naturally about the regions on a sphere.

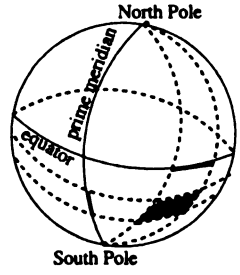


Figure 2: Example of a spherical rectangle.

There are 16 atomic relations between c_interval (o, s, d, fi, eq, f, di, si, oi, moi, ooi, mi, mmi, omi, ppi, m). Each one expresses a possible relation between the ends of two arcs in the closed curve. We have proved that all atomic relations are definable by means of the relation *meets* and the fundamental operations of composition, inversion and intersection. Consequently, the only predicate of the language of the logic of arcs is the predicate m : "x meets y". The interpretation of this predicate leads us to consider, as models of the logic of arcs, the set of the relational structures of the form $(Arcs, m)$ where the relation m which is binary on the set $Arcs$ of the arcs of a circle is the relation of meeting. Is the axiomatization of the set of the valid formulas that this class of models determines workable ? Is the logic of arcs decidable ? Is it complete ? Does it admit elimination of quantifiers ? It is not beyond the bounds of possibility that, drawing our inspiration from the axioms for time intervals [5], we become capable of putting forward the solutions of these problems about which we know nothing, all the more so since, as far as we are aware, no one has considered such issues. It is claimed, however, that the possible applications of the logic of arcs are numerous.

Example 1 *We should consider, for instance, the problems faced by firms which want to be able to run round the clock. Their personnel officers have to organise the timetable of many workers and must argue about the question of the relations between hours of work that periodically reoccur. Take another example : breaking up the horizon into its component parts, a robot sees 1-dimensional objects and has to reason about the issue of the links between portions of space that fill the skyline.*

There is strong probability that, inspired by the propagation techniques developed to solve the networks of constraints between intervals [1, 11, 7, 9, 12], the following model for reasoning about topologic relations between arcs is capable of tackling such problems. Adapting the method worked out in several articles on interval algebra, we merely present arc algebra as a set of relations, the arc relations, together with the fundamental operations of composition, inversion and intersection. Afterwards, encoding every interval network into some arc network, we show that the problem of proving the consistency of arc networks is NP-complete. Finally, we consider the related issue of how to demonstrate that nice networks are consistent.

2 Relations

Let $\mathcal{C}(C, r)$ be the circle with center C and radius r, $\mathcal{D}(C, r)$ be the set of the points situated inside it apart from C and $\mathcal{A}(C, r)$ be the set of its arcs. The first thing that needs to be said is that, according to the figure 3, there are 16 atomic relations which constitute the exhaustive list of the relations which can hold between the ends of two arcs $x, y \in \mathcal{A}(C, r)$: o, s, d, fi, eq, f, di, si, oi, moi, ooi, mi, mmi, omi, ppi and m. In particular, given the arc $x \in \mathcal{A}(C, r)$, there exists exactly one arc $y \in \mathcal{A}(C, r)$ such that $mmi(x, y)$, which will be represented by the expression x'. What is more, these atomic relations are definable by means of the relation m and the fundamental operations of composition, inversion and intersection.

Example 2 *For example,* $mmi = (m \circ m^{-1} \circ m) \cap (m^{-1} \circ m \circ m^{-1})$. *Take another example :* $si = mmi \circ m$. *Likewise,* $ppi = m^{-1} \circ si$.

Defining the dimension $\dim(A)$ of an atomic relation A as 2 minus the number of coincident endpoints that A determines, it may be mentioned in passing that there are 2 atomic relations of dimension 0 : eq and mmi, 8 atomic relations of dimension 1 : s, fi, f, si, moi, mi, omi and m and 6 atomic relations of dimension 2 : o, d, di, oi, ooi and ppi. Next we wish to focus our attention on the connection between these 16 atomic relations. Given the arc $x \in \mathcal{A}(C, r)$, let $y \in \mathcal{A}(C, r)$

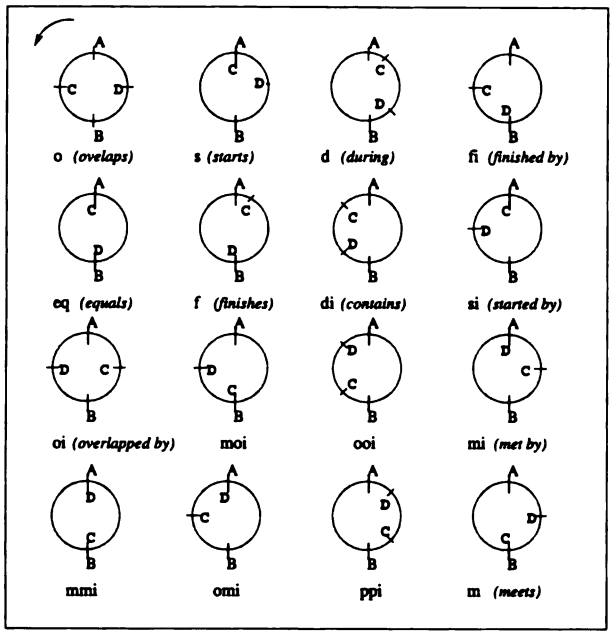


Figure 3: The permitted relations between the ends $x^- = A$, $x^+ = B$, $y^- = C$ and $y^+ = D$ of two arcs x and y.

be an arc linked to x by the relation o. While y^- moves clockwise, y turns into an arc linked to x by the relation s and after that by the relation d and while y^- moves counterclockwise, y turns into an arc linked to x by the relation m and after that by the relation ppi. Similarly, given the arc $x \in \mathcal{A}(C, r)$, let $y \in \mathcal{A}(C, r)$ be an arc linked to x by the relation oi. While y^+ moves clockwise, y turns into an arc linked to x by the relation mi and after that by the relation ppi and while y^+ moves counterclockwise, y turns into an arc linked to x by the relation f and after that by the relation d. This line of reasoning brings us to arrange the 16

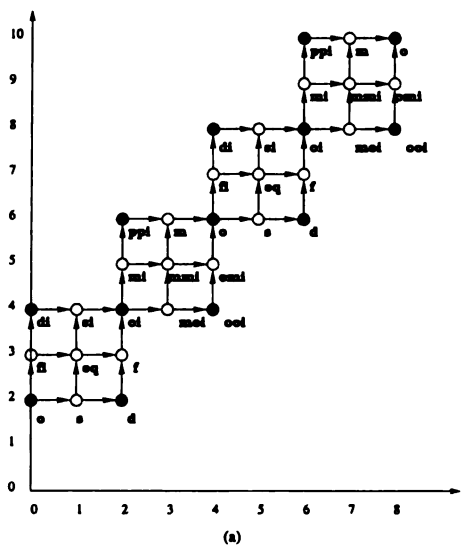


Figure 4: The graph of the atomic relations.

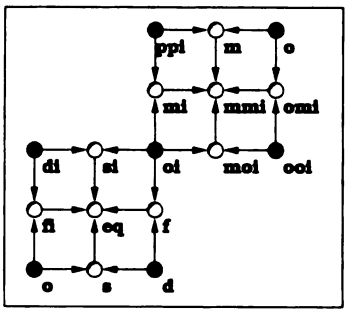


Figure 5: The Topologic closure of the atomic relations.

atomic relations in accordance with the figure 4 where an horizontal arrow between two nodes labelled with the atomic relations A and B signifies that, given the arc $x \in \mathcal{A}(C, r)$, while y^- moves clockwise, the arc $y \in \mathcal{A}(C, r)$ linked to x by the relation A turns into an arc linked to x by the relation B and a vertical arrow between two nodes labelled with the atomic relations A and B signifies that, given the arc $x \in \mathcal{A}(C, r)$, while y^+ moves clockwise, the arc $y \in \mathcal{A}(C, r)$ linked to x by the relation A turns into an arc linked to x by the relation B. Also, following the line of reasoning suggested by Ligozat [7], the topologic closure $C(A)$ of an atomic relation A is the set of the atomic relations that the figure 5 determines.

Example 3 *For example,* $C(mmi) = \{mmi\}$. *Take another example :* $C(si) = \{eq, si\}$. *Likewise,* $C(ppi) = \{mi, mmi, ppi, m\}$.

The arc algebra is made of a set of relations, the arc relations, which are sets of atomic relations. For every arc relation R, let $\dim(R) = \max\{\dim(A) : A \in R\}$ and $C(R) = \bigcup\{C(A) : A \in R\}$. What we are mainly concerned with here is the concept of convexity. In accordance with the figure 4, a convex relation is an arc relation which corresponds to the Cartesian product of two convex subsets of $\mathbb{N}$.

Example 4 *We should consider, for instance, the arc relation* $\{eq, f, si, oi, moi, mi, mmi\}$ *which is a convex relation, in accordance with its correspondence to the Cartesian product of the convex subsets* $\{1, 2, 3\}$ *and* $\{3, 4, 5\}$ *of* $\mathbb{N}$. *Take another example : the arc relation* $\{ppi, m, o, s, d\}$ *is a convex relation, seeing that it corresponds to the Cartesian product of the convex subsets* $\{2, 3, 4, 5, 6\}$ *and* $\{6\}$ *of* $\mathbb{N}$.

The arc algebra contains exactly 236 convex relations. What is more, adapting the concept of preconvexity introduced by Ligozat [7], a preconvex relation is an arc relation which topologic closure is convex.

Example 5 *We should consider, by way of illustration, the arc relation $\{moi, m\}$ which is a preconvex relation given that its topologic closure is the convex relation $\{moi, mmi, m\}$. Take another example : the arc relation $\{fi, f, di, oi\}$ is a preconvex relation for the simple reason that its topologic closure is the convex relation $\{fi, eq, f, di, si, oi, moi, mi, mmi\}$.*

In addition, a nice relation is either a preconvex relation of dimension 0 or 1 or a convex relation of dimension 2 containing at the very most 2 atomic relations. The arc algebra contains exactly 49 nice relations.

3 Areas

It is undeniably true that, for every arc $x \in \mathcal{A}(C,r)$, the ends of which are the points $x^- = A$ and $x^+ = B$ of the circle $\mathcal{C}(C,r)$, there exists exactly one point $X \in \mathcal{D}(C,r)$ such that X is situated inside the segment $[x^-, C]$ and $length(x) = 2\pi \times length([x^-, X])$. What is more, for every point $X \in \mathcal{D}(C,r)$, there exists exactly one arc $x \in \mathcal{A}(C,r)$ such that X is situated inside the segment $[x^-, C]$ and $length(x) = 2\pi \times length([x^-, X])$. All this goes to show that

Lemma 1 *The mapping ϕ which associates the point $X \in \mathcal{D}(C,r)$ to the arc $x \in \mathcal{A}(C,r)$ as above is a one-to-one relationship between $\mathcal{A}(C,r)$ and $\mathcal{D}(C,r)$.*

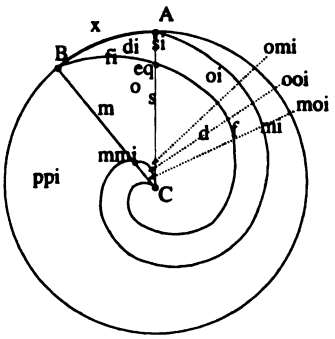


Figure 6: The ends $x^- = A$ and $x^+ = B$ of x determine areas inside $\mathcal{C}(C,r)$.

It is well worth noting that, for every arc $x \in \mathcal{A}(C,r)$, with endpoints $x^- = A$ and $x^+ = B$, $length([x^-, \phi(x)]) = length([\phi(x'), C])$ and, moreover, the spiral of Archimedes going clockwise from x^+ to C in exactly one revolution contains $\phi(x)$ whereas the spiral of Archimedes going clockwise from x^- to C in exactly one revolution contains $\phi(x')$. This one-to-one relationship raises the whole question of whether, for every arc $x \in \mathcal{A}(C,r)$, the endpoints of which are $x^- = A$ and $x^+ = B$, one can divide the set $\mathcal{D}(C,r)$ of the points situated inside $\mathcal{C}(C,r)$ apart from C into 16 pieces with regard to the 16 permitted relations between the arcs of $\mathcal{C}(C,r)$ and x.
We must do it so that a point situated inside $\mathcal{C}(C,r)$ apart from C belongs to a piece of the division iff the arc of $\mathcal{C}(C,r)$ associated with the point is linked to x by the relation associated with the piece. The truth of the matter is that x determines 16 areas inside $\mathcal{C}(C,r)$ according to the figure 6 where the area inside $\mathcal{C}(C,r)$ determined by the atomic relations fi, eq and f is the spiral of Archimedes going clockwise from x^+ to C in exactly one revolution and the area inside $\mathcal{C}(C,r)$ determined by the atomic relations mi, mmi and omi is the spiral of Archimedes going clockwise from x^- to C in exactly one revolution. For every atomic relation A, let $area(x,A)$ be the area inside $\mathcal{C}(C,r)$ determined by A. Let us remember that, for every arc $y \in \mathcal{A}(C,r)$, $A(x,y)$ iff $\phi(y) \in area(x,A)$. For every arc relation R, let $area(x,R) = \bigcup\{area(x,A) : A \in R\}$ be the area inside $\mathcal{C}(C,r)$ determined by R. It should never be forgotten that, for every arc $y \in \mathcal{A}(C,r)$, $R(x,y)$ iff $\phi(y) \in area(x,R)$. A number of key issues arise from the particular shape of the areas inside $\mathcal{C}(C,r)$ determined by convex relations.

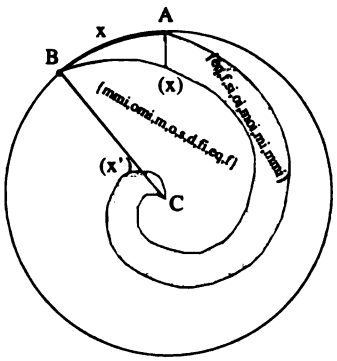


Figure 7: Areas inside $\mathcal{C}(C,r)$ given by the spiral movement of segments aligned with C.

According to the figure 7, take the case of the area inside $\mathcal{C}(C,r)$ determined by the relation $\{eq, f, si, oi, moi, mi, mmi\}$ which can be produced while the segment $[x^-, \phi(x)]$ spirally moves up to the segment $[\phi(x'), C]$ and consider similarly the area inside $\mathcal{C}(C,r)$ determined by the relation $\{mmi, omi, m, o, s, d, fi, eq, f\}$ which can be generated by the spiral movement of the segment $[x^+, \phi(x')]$. The inescapable conclusion which emerges from what we have said is that the 236 convex relations determine areas inside $\mathcal{C}(C,r)$ which can be given by the spiral movement of segments aligned with C.

	o	s	d	fi	f	di	si	oi	moi	ooi	mi	mmi	omi	ppi	m
o	oi,moi,ooi mi,mmi,o omi,ppi,m	ooi omi o	ooi,omi o,s,d	ppi,m o	o,s,d	ppi,m o,fi,di	o,fi di	o,s,d,fi eq,f,di si,oi	d,f oi	d,f,oi moi,ooi	di,si oi	oi	oi moi ooi	di,si,oi mi,ppi	oi,mi ppi
s	ppi,m,o	s	d	ppi m,o	d	ppi,m,o fi,di	s,eq si	d,f,oi	d,f oi	d,f,oi moi,ooi	mi	mi	mi,mmi omi	ppi	ppi
d	ppi,m,o s,d	d	d	ppi m,o s,d	d	ppi,m,o s,d,fi,eq f,di,si,oi mi	d,f,oi mi,ppi	d,f,oi,mi ppi	d,f oi,mi ppi	d, f,oi,moi ooi,mi,mmi omi ppi,m,o,s	ppi	ppi	ppi m,o s,d	ppi	ppi
fi	ooi,omi o	ooi omi,o	ooi,omi o,s,d	fi	fi,eq f	di	di	di,si,oi	moi	ooi	di,si oi	moi	ooi	di,si,oi mi,ppi	moi mmi,m
f	o,s,d	d	d	fi,eq f	f	di,si,oi mi,ppi	oi,mi ppi	oi,mi ppi	moi mmi,m	ooi,omi o,s,d	ppi	m	o,s,d	ppi	m
di	ooi,omi o,fi,di	ooi omi,o fi,di	ooi,omi,o s,d,fi,eq f,di,si oi,moi	di	di,si oi moi ooi	di	di	di,si,oi moi,ooi	ooi	ooi	di,si oi moi ooi	ooi	ooi	di,si,oi,moi ooi,mi,mmi omi,ppi m,o,fi	ooi omi o,fi,di
si	o,fi,di	s,eq si	d,f,oi moi,ooi	di	oi,moi ooi	di	si	oi,moi ooi	ooi	ooi	mi,mmi omi	omi	omi	ppi,m,o fi,di	o,fi,di
oi	o,s,d,fi eq,f,di si,oi	d,f oi	d,f,oi moi,ooi	di,si oi	oi moi ooi	di,si,oi mi,ppi	oi mi ppi	oi,moi,ooi mi,mmi,o omi,ppi,m	ooi,omi o	ooi,omi o,s,d	ppi m,o	o	o,s,d	ppi,m,o fi,di	o,fi,di
moi	di,si,oi	moi	ooi	di,si oi	ooi	di,si,oi mi,ppi	moi mmi,m	ooi,omi o	ooi omi,o	ooi,omi o,s,d	fi	fi	fi,eq f	di	di
ooi	di,si,oi moi,ooi	ooi	ooi	di,si oi,moi ooi	ooi	di,si,oi,moi ooi,mi,mmi omi,ppi m,o,fi	ooi omi o,fi,di	ooi,omi o,fi,di	ooi,omi o,fi,di	ooi,omi,o s d,fi,eq,f di,si,oi,moi	di	di	di,si,oi moi ooi	di	di
mi	d,f,oi	d,f oi	d,f,oi moi,ooi	mi	mi,mmi omi	ppi	ppi	ppi,m,o	s	d	ppi m,o	s	d	ppi,m,o fi,di	s,eq,si
mmi	oi	moi	ooi	mi	omi	ppi	m	o	s	d	fi	eq	f	di	si
omi	oi,moi ooi	ooi	ooi	mi,mmi omi	omi	ppi,m,o fi,di	o,fi di	o,fi,di	s,eq si	d,f,oi moi,ooi	di	si	oi,moi ooi	di	si
ppi	d,f,oi,mi ppi	d,f oi mi ppi	d,f,oi,moi ooi,mmi mi,omi,ppi m,o,s	ppi	ppi m,o s,d	ppi	ppi	ppi,m,o s,d	d	d	ppi m,o s,d	d	d	ppi,m,o s,d,fi,eq f,di,si,oi mi	d,f,oi mi,ppi
m	oi,mi ppi	moi mmi,m	ooi,omi o,s,d	ppi	o,s d	ppi	m	o,s,d	d	d	fi,eq f	f	f	di,si,oi mi,ppi	oi,mi ppi

Figure 8: The table of composition.

4 Fundamental operations

Composition of atomic relations is defined through the figure 8. It should never be forgotten that, for every atomic relation A, B and for every arc $x, y \in \mathcal{A}(C, r)$, there exists an atomic relation $C \in A \circ B$ such that $C(x, y)$ iff there exists an arc $z \in \mathcal{A}(C, r)$ such that $A(x, z)$ and $B(z, y)$. Composition of arc relations is defined in the following way, for every arc relation R, S : $R \circ S = \bigcup\{A \circ B : A \in R,\ B \in S\}$. Inversion of atomic relations is defined through the figure 9. Let

A	o	s	d	eq	f	moi	ooi	mmi	ppi	m
A^{-1}	oi	si	di	eq	fi	omi	ooi	mmi	ppi	mi

Figure 9: The table of inversion.

us remember that, for every atomic relation A and for every arc $x, y \in \mathcal{A}(C, r)$, $A^{-1}(x, y)$ iff $A(y, x)$. Inversion of arc relations is defined in the following way, for every arc relation R : $R^{-1} = \bigcup\{A^{-1} : A \in R\}$. The difficulty about proving the

Lemma 2 *For every convex relation R, S, the arc relation $R \circ S$ is a convex relation.*

and

Lemma 3 *For every convex relation R, the arc relation R^{-1} is a convex relation.*

is that we really ought to perform a great number of calculations without commiting any error. To this end we have written a program capable of automatically proving that the concept of convexity is preserved by the fundamental operations of composition and inversion. An added result that our program is capable of automatically demonstrating is that, for every atomic relation A, B, $C(A \circ B) \supseteq C(A) \circ C(B)$ and, for every atomic relation A, $C(A^{-1}) = C(A)^{-1}$. Consequently, drawing our inspiration from the line of reasoning suggested by Ligozat [7], we must conclude that

Lemma 4 *For every arc relation R, $C(R^{-1}) = C(R)^{-1}$.*

and

Lemma 5 *For every arc relation R, S, $C(R \circ S) \supseteq C(R) \circ C(S)$.*

On the other hand, one instance is enough to show that the concept of convexity is not preserved by the fundamental operation of intersection. We should consider, for example, the convex relation $\{d, f, oi, moi, ooi\}$ together with the convex relation $\{ooi, omi, o, s, d\}$. Though there is still some doubt surrounding the

preservation of the concept of preconvexity by the fundamental operation of composition, it must be stressed that

Lemma 6 *For every preconvex relation R, the arc relation R^{-1} is a preconvex relation.*

Proof This is a direct consequence of the lemmas 3 and 4.
About the preservation of the concept of nicety, the most important thing is to remark that

Lemma 7 *For every nice relation R, S, T, the arc relation $R \cap (S \circ T)$ is a nice relation.*

Proof Direct calculations.

5 Network

It is now time to adapt the propagation techniques designed to solve the networks of constraints between intervals [1, ?, 7, 9, 12]. An arc network is a structure

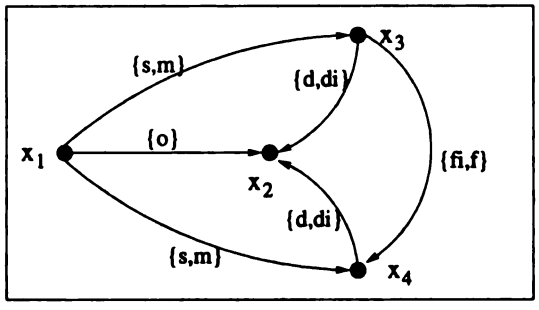


Figure 10: Example of a path-consistent arc network which is not consistent.

of the form (n, R) where $n \geq 1$ and R is a mapping of $(n) \times (n)$ to the set of the arc relations such that, for every $i \in (n)$, $R_{i,i} = \{eq\}$ and, for every $i, j \in (n)$, $R_{j,i} = R_{i,j}^{-1}$. In the first place, an important aspect is the property of consistency of the arc network (n, R) which means that there exists a mapping x of (n) to $\mathcal{A}(C, r)$ such that, for every $i, j \in (n)$, $R_{i,j}(x_i, x_j)$. Second, let us consider the property of path-consistency of the arc network (n, R) which means that, for every $i, j \in (n)$, $R_{i,j} \neq \emptyset$ and, for every $i, j, k \in (n)$, $R_{i,j} \subseteq R_{i,k} \circ R_{k,j}$. What should be established at the very outset is that

Lemma 8 *For every arc network (n, R), if (n, R) is consistent then it can be transformed into a path-consistent arc network by means of the path-consistency algorithm which consists in successively replacing, for every $i, j, k \in (n)$, the constraint $R_{i,j}$ by the constraint $R_{i,j} \cap (R_{i,k} \circ R_{k,j})$.*

The converse, however, is not true, seeing that there exists path-consistent arc networks which are not consistent. To illustrate the truth of this, one has only to mention the arc network of the figure 10. In actual fact,

Lemma 9 *The problem of proving the consistency of arc networks is NP-hard.*

Proof The fact of the matter is that the NP-complete problem of proving the consistency of interval networks is polynomial-time reducible to the problem of proving the consistency of arc networks.
On the other hand, it is observable that

Lemma 10 *The problem of proving the consistency of arc networks is in NP.*

Proof Given an arc network (n, R), both the selection at random of a mapping x of (n) to $\mathcal{A}(C, r)$ and the proof that, for every $i, j \in (n)$, $R_{i,j}(x_i, x_j)$ can be done in polynomial time by a nondeterministic Turing machine.
Ultimately, then,

Theorem 1 *The problem of proving the consistency of arc networks is NP-complete.*

All this may well be true enough, but we should formulate the following problem : find a set of arc networks which consistency can be decided in polynomial time by means of the path-consistency algorithm. In this respect, it is interesting to consider the property of nicety of the arc network (n, R) which means that, for every $i, j \in (n)$, $R_{i,j}$ is a nice relation and, for every $i, j, k \in (n)$, if $i \neq j$, $i < k$ and $j < k$ then either $R_{i,k}$ is a preconvex relation of dimension 0 or 1 or $R_{j,k}$ is a preconvex relation of dimension 0 or 1. It is important to be clear that

Lemma 11 *The concept of nicety is preserved by the path-consistency algorithm.*

Proof This is a direct consequence of the lemma 7.

6 Tractability

Let us now consider to what extent the problem of proving the consistency of nice networks can be decided by means of the path-consistency algorithm. First of all, let us begin with an examination of the

Lemma 12 *For every arc $x_1, \ldots, x_K$ and for every preconvex relation $R_1, \ldots, R_K$ of dimension 0 or 1, if, for every $i, j \in (K)$, $area(x_i, R_i) \cap area(x_j, R_j) \neq \emptyset$ then $\bigcap_{k \in (K)} area(x_k, R_k) \neq \emptyset$.*

Proof In the first place, assume that (1) there exists $l \in (K)$ such that $R_l = \{eq\}$. In this case, $area(x_l, R_l) = \{\phi(x_l)\}$ and, for every $k \in (K)$, using the hypothesis that, $area(x_k, R_k) \cap area(x_l, R_l) \neq \emptyset$, one can deduce that $\phi(x_l) \in area(x_k, R_k)$. Consequently, $\bigcap_{k\in(K)} area(x_k, R_k) = \{\phi(x_l)\}$.
In the second place, suppose that (2) there exists $l \in (K)$ such that $R_l = \{mmi\}$. There is not much to choose between this case and the case (1).
In the third place, assume that, (3) for every $l \in (K)$, $R_l \neq \{eq\}$ and $R_l \neq \{mmi\}$. Firstly, put the case that, (3.1) for every $l \in (K)$, $R_l \subseteq \{s, eq, si\}$ or $R_l \subseteq \{moi, mmi, m\}$. Accordingly, using the hypothesis that, for every $i, j \in (K)$, $area(x_i, R_i) \cap area(x_j, R_j) \neq \emptyset$, one can prove that there exists a radius of $\mathcal{C}(C, r)$ which contains $area(x_k, R_k)$, for every $k \in (K)$, and, applying the theorem of Helly [3] on this radius, one can deduce that $\bigcap_{k\in(K)} area(x_k, R_k) \neq \emptyset$.
Secondly, put the case that, (3.2) for every $l \in (K)$, $R_l \subseteq \{fi, eq, f\}$ or $R_l \subseteq \{mi, mmi, omi\}$. There is essentially no difference between this case and the case (3.1).
Thirdly, put the case that (3.3) there exists $i, j \in (K)$ such that $R_i \subseteq \{s, eq, si\}$ or $R_i \subseteq \{moi, mmi, m\}$ and $R_j \subseteq \{fi, eq, f\}$ or $R_j \subseteq \{mi, mmi, omi\}$. Accordingly, using the hypothesis that $area(x_i, R_i) \cap area(x_j, R_j) \neq \emptyset$, one can deduce that there exists $X \in \mathcal{D}(C, r)$ such that $area(x_i, R_i) \cap area(x_j, R_j) = \{X\}$. Seeing that, for every $k \in (K)$, $X \in area(x_k, R_k)$, we must conclude that $\bigcap_{k\in(K)} area(x_k, R_k) = \{X\}$.Likewise,

Lemma 13 *For every arc x, for every arc $x_1, \ldots, x_K$, for every convex relation R of dimension 2 containing at the very most 2 atomic relations and for every preconvex relation $R_1, \ldots, R_K$ of dimension 0 or 1, if, for every $i \in (K)$, $area(x, R) \cap area(x_i, R_i) \neq \emptyset$ and, for every $i, j \in (K)$, $area(x_i, R_i) \cap area(x_j, R_j) \neq \emptyset$ then $area(x, R) \cap \bigcap_{k\in(K)} area(x_k, R_k) \neq \emptyset$.*

This, of course, leads to the logical conclusion that

Theorem 2 *The problem of proving the consistency of nice networks can be decided in polynomial time by means of the path-consistency algorithm.*

Proof According to the lemma 11, the concept of nicety is preserved by the path-consistency algorithm. Consequently, the issue at stake here is the demonstration that every path-consistent nice network is consistent. First of all, let us consider a path-consistent nice network (n, R). Given $K \in (n-1)$, let us now look at a mapping x of (K) to $\mathcal{A}(C, r)$ such that, for every $i, j \in (K)$, $R_{i,j}(x_i, x_j)$. Therefore, seeing that (n, R) is path-consistent, for every $i, j \in (K)$, there exists an arc x_{K+1} such that $R_{i,K+1}(x_i, x_{K+1})$ and $R_{j,K+1}(x_j, x_{K+1})$. This only goes to show that, for every $i, j \in (K)$, $area(x_i, R_{i,K+1}) \cap area(x_j, R_{j,K+1}) \neq \emptyset$ and, according to the lemmas 12 and 13, $\bigcap_{k\in(K)} area(x_k, R_{k,K+1}) \neq \emptyset$. In conclusion, there exists an arc x_{K+1} such that, for every $k \in (K)$, $R_{k,K+1}(x_k, x_{K+1})$.
The network of the figure 11 serves to illustrate this tractability result.

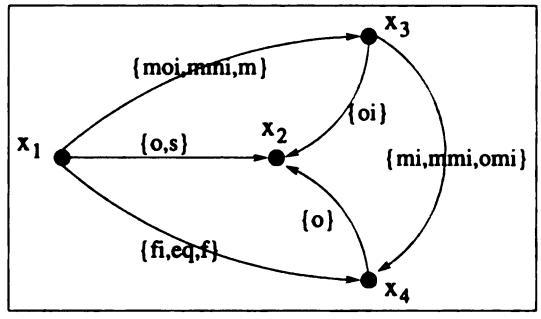


Figure 11: Example of a path-consistent nice network.

7 Conclusion

In this paper, we have expounded arc algebra as a set of relations, the arc relations, together with the fundamental operations of composition, inversion and intersection. Also, we have introduced the concept of convexity just as the one of preconvexity and we have proved that the concept of convexity is preserved by the fundamental operation of composition whereas we have left open the question of the preservation of the concept of preconvexity. In the same way, we have shown that every one of these concepts are preserved by the fundamental operation of inversion. In conclusion, fitting the propagation techniques developed to solve the networks of constraints between intervals to the NP-complete problem of proving the consistency of arc networks, we have demonstrated that the problem of proving the consistency of nice networks can be decided in polynomial time by means of the path-consistency algorithm.

References

[1] J.F. Allen. Maintaining knowledge about temporal intervals. *Journal of the Association for Computing Machinery*, 26:832–843, 1983.

[2] P. Balbiani, A. Osmani, J. Condotta, and L. Fariñas del Cerro. Reasoning about generalized intervals. In *Lecture Notes in Artificial Intelligence(1480)*, pages 50–61. Springer, International

Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA-98), 1998.

[3] V. Chvátal. *Linear Programming.* Freeman, 1983.

[4] Z. Cui, A.G. Cohn, and D.A. Randell. Qualitative simulation based on a logical formalism of space and time. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 679–684, 1992.

[5] P. Ladkin. Models of axioms for time intervals. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 234–239, 1987.

[6] G. Ligozat. On generalized interval calculi. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 234–240, 1991.

[7] G. Ligozat. A new proof of tractability for ORD-HORN relations. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 395–401, 1996.

[8] R.A. Morris, W.D. Shoaff, and L. Khatib. Path consistency in a network of non-convex intervals. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 655–660, 1993.

[9] B. Nebel and H.J. Bürckert. Reasoning about temporal relations: a maximal tractable subset of Allen's interval algebra. *Journal of the Association for Computing Machinery*, 42:43–66, 1995.

[10] J. Renz and B. Nebel. Spatial reasoning with topological information. *Lecture Notes in Computer Science*, 1404:351–370, 1998.

[11] P. van Beek. Approximation algorithms for temporal reasoning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1291–1296, 1989.

[12] M. Vilain and H. Kautz. Constraint propagation algorithms for temporal reasoning. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 377–382, 1986.

Automated Reasoning II

Partition-Based Logical Reasoning

Eyal Amir and **Sheila McIlraith***
Department of Computer Science,
Gates Building, 2A wing
Stanford University, Stanford, CA 94305-9020, USA
{eyal.amir,sheila.mcilraith}@cs.stanford.edu

Abstract

We investigate the problem of reasoning with partitions of related logical axioms. Our motivation is two-fold. First, we are concerned with how to reason effectively with multiple knowledge bases that have overlap in content. Second, and more fundamentally, we are concerned with how to exploit structure inherent in a set of logical axioms to induce a partitioning of the axioms that will lead to an improvement in the efficiency of reasoning. To this end, we provide algorithms for reasoning with partitions of axioms in propositional and first-order logic. Craig's interpolation theorem serves as a key to proving completeness of these algorithms. We analyze the computational benefit of our algorithms and detect those parameters of a partitioning that influence the efficiency of computation. These parameters are the number of symbols shared by a pair of partitions, the size of each partition, and the topology of the partitioning. Finally, we provide a greedy algorithm that automatically decomposes a given theory into partitions, exploiting the parameters that influence the efficiency of computation.

1 Introduction

There is growing interest in building large knowledge bases (KBs) of everyday knowledge about the world, teamed with theorem provers to perform inference. Three such systems are Cycorp's Cyc, and the High Performance Knowledge Base (HPKB) systems developed by Stanford's Knowledge Systems Lab (KSL) [23] and by SRI (e.g., [12]). These KBs comprise tens/hundreds of thousands of logical axioms. One approach to dealing with the size and complexity of these KBs is to structure the content in some way, such as into multiple domain- or task-specific KBs, or into microtheories. In this paper, we investigate how to reason effectively with partitioned sets of logical axioms that have overlap in content, and that may even have different reasoning engines. More generally, we investigate the problem of how to exploit structure inherent in a set of logical axioms to induce a partitioning of the axioms that will improve the efficiency of reasoning.

To this end, we propose *partition-based* logical reasoning algorithms, for reasoning with logical theories[1] that are decomposed into related partitions of axioms. Given a partitioning of a logical theory, we use Craig's interpolation theorem [15] to prove the soundness and completeness of a forward message-passing algorithm and an algorithm for propositional satisfiability. The algorithms are designed so that, without loss of generality, reasoning within a partition can be realized by an arbitrary consequence-finding engine, in parallel with reasoning in other partitions. We investigate the impact of these algorithms on resolution-based inference, and analyze the computational complexity for our partition-based SAT.

A critical aspect of partition-based logical reasoning is the selection of a *good* partitioning of the theory. The computational analysis of our partition-based reasoning algorithms provides a metric for identifying parameters of partitionings that influence the computation of our algorithms: the *bandwidth* of communication between partitions, the size of each partition, and the topology of the partitions graph. These parameters guide us to propose a greedy algorithm for decomposing logical theories into partitions, trying to optimize these parameters.

Surprisingly, there has been little work on the specific problem of exploiting structure in theorem proving and SAT in the manner we propose. This can largely be attributed to the fact that theorem proving has traditionally examined mathematics domains, that do not necessarily have structure that supports decomposition. Nevertheless, there are

*Knowledge Systems Laboratory (KSL)

[1] In this paper, every set of axioms is a *theory* (and vice versa).

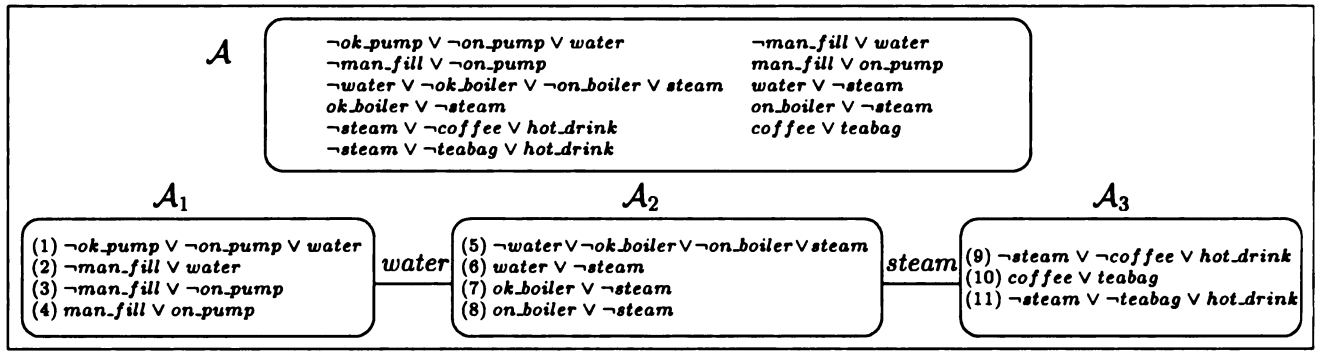


Figure 1: A partitioning of $\mathcal{A}$ and its intersection graph.

many areas of related work, which we discuss at the end of this paper.

2 Partition-Based Theorem Proving

In this section we address the problem of how to reason with an already partitioned propositional or first-order logic (FOL) theory. In particular, we propose a forward message-passing algorithm, in the spirit of Pearl [40], and examine the effect of this algorithm on resolution-based inference.

$\{\mathcal{A}_i\}_{i\leq n}$ is a *partitioning* of a logical theory $\mathcal{A}$ if $\mathcal{A} = \bigcup_i \mathcal{A}_i$. Each individual $\mathcal{A}_i$ is called a *partition*, and $\mathcal{L}(\mathcal{A}_i)$ is its signature (the non-logical symbols). Each such partitioning defines a labeled graph $G = (V, E, l)$, which we call the *intersection graph*. In the intersection graph, each node i represents an individual partition $\mathcal{A}_i$, ($V = \{1, ..., n\}$), two nodes i, j are linked by an edge if $\mathcal{L}(\mathcal{A}_i)$ and $\mathcal{L}(\mathcal{A}_j)$ have a symbol in common ($E = \{(i,j) \mid \mathcal{L}(\mathcal{A}_i) \cap \mathcal{L}(\mathcal{A}_j) \neq \emptyset\}$), and the edges are labeled with the set of symbols that the associated partitions share ($l(i,j) = \mathcal{L}(\mathcal{A}_i) \cap \mathcal{L}(\mathcal{A}_j)$). We refer to $l(i,j)$ as the *communication language* between partitions $\mathcal{A}_i$ and $\mathcal{A}_j$. We ensure that the intersection graph is connected by adding a minimal number of edges to E with empty labels, $l(i,j) = \emptyset$.

We illustrate the notion of a partitioning in terms of the simple propositional theory $\mathcal{A}$, depicted at the top of Figure 1. (This is the clausal form of the theory presented with material implication in Figure 2.) This set of axioms captures the functioning of aspects of an expresso machine. The top four axioms denote that if the machine pump is OK and the pump is on then the machine has a water supply. Alternately, the machine can be filled manually, but it is never the case that the machine is manually filling while the pump is on. The second four axioms denote that there is steam if and only if the boiler is OK and is on, and there is a supply of water. Finally, there is always either coffee or tea. Steam and coffee (or tea) result in a hot drink.

$ok\_pump \wedge on\_pump \Rightarrow water$	$man\_fill \Rightarrow water$
$man\_fill \Rightarrow \neg on\_pump$	$\neg man\_fill \Rightarrow on\_pump$
$water \wedge ok\_boiler \wedge on\_boiler \Rightarrow steam$	$\neg water \Rightarrow \neg steam$
$\neg ok\_boiler \Rightarrow \neg steam$	$\neg on\_boiler \Rightarrow \neg steam$
$steam \wedge coffee \Rightarrow hot\_drink$	$coffee \vee teabag$
$steam \wedge teabag \Rightarrow hot\_drink$	

Figure 2: Axiomatization of a simplified espresso machine.

The bottom of Figure 1 depicts a decomposition of $\mathcal{A}$ into three partitions $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ and its intersection graph. The labels for the edges $(1,2), (2,3)$ are $\{water\}$ and $\{steam\}$, respectively.

2.1 Forward Message Passing

In this section we propose a forward message-passing algorithm for reasoning with partitions of logical axioms. Figure 3 describes our forward message-passing algorithm, FORWARD-M-P (MP) for finding the truth value of query formula Q whose signature is in $\mathcal{L}(\mathcal{A}_k)$, given partitioned theory $\mathcal{A}$ and graph $G = (V, E, l)$, possibly the intersection graph of $\mathcal{A}$, but not always so.

PROCEDURE FORWARD-M-P($\{\mathcal{A}_i\}_{i\leq n}, G, Q$)
$\{\mathcal{A}_i\}_{i\leq n}$ a partitioning of the theory $\mathcal{A}$, $G = (V, E, l)$ a graph describing the connections between the partitions, Q a query formula in the language of $\mathcal{L}(\mathcal{A}_k)$ ($k \leq n$).

1. Let $dist(i,j)$ ($i, j \in V$) be the length of the shortest path between i, j in G. Let $i \prec j$ iff $dist(i,k) < dist(j,k)$ ($\prec$ is a strict partial order).
2. Concurrently perform consequence finding for each of the partitions $\mathcal{A}_i$, $i \leq n$.
3. For every $(i,j) \in E$ such that $i \prec j$, if we prove $\mathcal{A}_j \models \varphi$ and φ's signature is in $\mathcal{L}(l(i,j))$, then add φ to the set of axioms of $\mathcal{A}_i$.
4. If we proved Q in $\mathcal{A}_k$, return YES.

Figure 3: A forward message-passing algorithm.

This algorithm exploits consequence finding (step 2) to per-

form reasoning in the individual partitions. Consequence finding was defined by Lee [33] to be the problem of finding all the logical consequences of a theory or sentences that subsume them. In MP, we can use any sound and complete consequence-finding algorithm. The *resolution rule* is complete for consequence finding (e.g., [33, 46]) and the same is true for *semantic resolution* [47] (and *set-of-support resolution* [24]), and *linear resolution* variants (e.g., [30]). Such consequence finders are used for prime implicate generation in applications such as diagnosis. Inoue [30] provides an algorithm for selectively generating consequences or *characteristic clauses* in a given sub-vocabulary. We can exploit this algorithm to focus consequence finding on axioms whose signature is in the communication language of the partition.

Figure 4 illustrates an execution of MP using resolution.

Using FORWARD-M-P to prove *hot_drink*

Part.	Resolve	Generating	
$\mathcal{A}_1$	(2) , (4)	*on_pump* ∨ *water*	(m1)
$\mathcal{A}_1$	(m1), (1)	*ok_pump* ∨ *water*	(m2)
$\mathcal{A}_1$	(m2), (12)	*water*	(m3)
		clause *water* passed from $\mathcal{A}_1$ to $\mathcal{A}_2$	
$\mathcal{A}_2$	(m3) , (5)	*ok_boiler* ∧ *on_boiler* ⊃ *steam*	(m4)
$\mathcal{A}_2$	(m4) , (13)	¬*on_boiler* ∨ *steam*	(m5)
$\mathcal{A}_2$	(m5) , (14)	*steam*	(m6)
		clause *steam* passed from $\mathcal{A}_2$ to $\mathcal{A}_3$	
$\mathcal{A}_3$	(9) , (10)	¬*steam* ∨ *teabag* ∨ *hot_drink*	(m7)
$\mathcal{A}_3$	(m7) , (11)	¬*steam* ∨ *hot_drink*	(m8)
$\mathcal{A}_3$	(m8) , (m6)	*hot_drink*	(m9)

Figure 4: A proof of *hot_drink* from $\mathcal{A}$ in Figure 1 after asserting *ok_pump* (12) in $\mathcal{A}_1$ and *ok_boiler* (13), *on_boiler* (14) in $\mathcal{A}_2$.

Given a partitioning whose intersection graph forms an *undirected tree*, our MP algorithm is a sound and complete proof procedure. The completeness relies on Craig's Interpolation Theorem.

Theorem 2.1 (Craig's Interpolation Theorem [15]) *If $\alpha \vdash \beta$, then there is a formula γ involving only symbols common to both α and β, such that $\alpha \vdash \gamma$ and $\gamma \vdash \beta$.*

Craig's interpolation theorem is true even if we take α, β to be infinite sets of sentences [46], and use resolution theorem proving [46, 29], with or without equality [15, 16] (all after proper reformulation of the theorem). Lyndon's version of the interpolation theorem [35] adds sensitivity to the polarity of relation symbols (γ includes P positively (negatively) only if P shows positively (negatively) in both α and β).

Theorem 2.2 (Soundness and Completeness) *Let $\mathcal{A} = \bigcup_{i \le n} \mathcal{A}_i$ be a partitioned theory with the intersection graph G being a tree (i.e., no cycles). Let $k \le n$ and φ a sentence whose signature is in $\mathcal{L}(\mathcal{A}_k)$. Then, $\mathcal{A} \models \varphi$ iff MP outputs YES.*

PROOF See Appendix A.1.

If the intersection graph of $\mathcal{A}$ is not a tree, and MP uses it as input, then MP may fail to be a complete proof procedure. Figure 5 illustrates the problem. The left-hand side of Figure 5 illustrates the intersection graph of partitioning $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ of a theory $\mathcal{A}$. If we try to prove s (which follows from $\mathcal{A}$) from this partitioning and graph using MP, nothing will be transmitted between the partitions. For example, we cannot send $p \Rightarrow s$ from $\mathcal{A}_2$ to $\mathcal{A}_4$ because the graph only allows transmission of sentences containing s.

Thus, using MP with the left-hand side graph will fail to prove s. In such a case, we must first syntactically transform the intersection graph into a tree with enlarged labels, (i.e. an enlarged communication language) and apply MP to the resultant tree. Algorithm BREAK-CYCLES, shown in Figure 6, performs the appropriate transformation ($|X|$ is the cardinality of a set X).

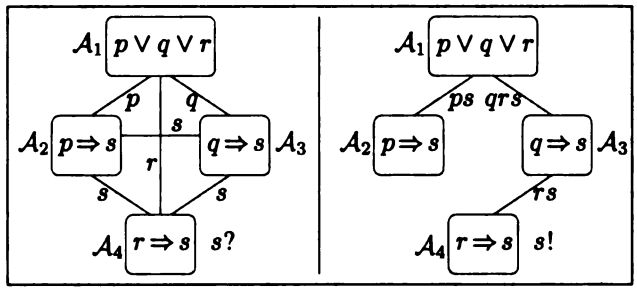


Figure 5: An intersection graph before (left) and after (right) applying BREAK-CYCLES.

PROCEDURE BREAK-CYCLES($G = (V, E, l)$)

1. Find a minimal-length cycle of nodes $i_1, ..., i_c$ in G. If there are no cycles, return G.
2. Select index a s.t. $a < c$ and $\sum_{a \neq j < c} |l(i_j, i_{j+1}) \cup l(i_a, i_{a+1})|$ is minimal (the label of (i_a, i_{a+1}) adds a minimal number of symbols to the rest of the cycle).
3. For all $j < c$, $j \neq a$, set $l(i_j, i_{j+1}) \leftarrow l(i_j, i_{j+1}) \cup l(i_a, i_{a+1})$.
4. Set $E \leftarrow E \setminus \{(i_a, i_{a+1})\}$, $l(i_a, i_{a+1}) \leftarrow \emptyset$ and go to 1.

Figure 6: An algorithm to transform an intersection graph G into a tree.

Using BREAK-CYCLES, we can transform the graph depicted on the left-hand side of Figure 5, into the

tree on its right. First, we identify the minimal cycle $\langle(1,3),(3,4),(4,1)\rangle$, remove $(4,1)$ from E and add r to the labels of $(1,3),(3,4)$. Then, we find the minimal cycle $\langle(2,3),(3,4),(4,2)\rangle$ and remove $(2,3)$ from E (s already appears in the labels of $(4,2),(3,4)$). Finally, we identify the minimal cycle $\langle(1,3),(3,4),(4,2),(2,1)\rangle$, remove $(4,2)$ and add s to the rest of the cycle. The proof of s by MP now follows by sending $p \Rightarrow s$ from $\mathcal{A}_2$ to $\mathcal{A}_1$, sending $q \vee r \vee s$ from $\mathcal{A}_1$ to $\mathcal{A}_3$, sending $r \vee s$ from $\mathcal{A}_3$ to $\mathcal{A}_4$ and concluding s in $\mathcal{A}_4$.

Notice that when executing BREAK-CYCLES, we may remove an edge that participates in more than one minimal cycle (as is the case of removing the edge $(4,1)$), but its removal influences the labels of only one cycle.

Theorem 2.3 (Soundness and Completeness) *Let $\mathcal{A} = \bigcup_{i \leq n} \mathcal{A}_i$ be a partitioned theory with the intersection graph G. Let $k \leq n$ and φ a sentence whose signature is in $\mathcal{L}(\mathcal{A}_k)$. $\mathcal{A} \models \varphi$ iff applying BREAK-CYCLES and then MP outputs YES.*

PROOF See Appendix A.2.

BREAK-CYCLES is a greedy algorithm that has a worst-case complexity of $O(|E|^2 * m)$ (where m is the number of symbols in $\mathcal{L}(\mathcal{A})$). An algorithm for finding a minimum spanning tree (e.g., [22]) can be used to compute the optimal subgraph for our purpose (minimizing $\sum_{(i,j)\in E} |l(i,j)|$) in time $O(n^2 + m * n)$, if we assume that the labels are mutually disjoint. It is not clear whether the algorithm can be generalized to provide an optimal solution to the case of arbitrary labels.

Note that MP requires that query Q be in the language of a single partition, $\mathcal{L}(\mathcal{A}_k)$. One way to answer a query Q that comprises symbols drawn from multiple partitions is to add a new partition $\mathcal{A}_Q$, with $\mathcal{L}(\mathcal{A}_Q) = \mathcal{L}(Q)$, the signature the query. $\mathcal{A}_Q$ may contain $\neg Q$ or no axioms. Following addition of this new partition, BREAK-CYCLES must be run on the new intersection graph. To prove Q in $\mathcal{A}_Q$, we run MP on the resulting graph.

Our MP algorithm uses the query Q to induce an ordering on the partitions, which in turn may guide selective consequence finding for *reasoning forward*. Many theorem proving strategies exploit the query more aggressively by *reasoning backwards* from the query. Such strategies have proven effective for a variety of reasoning problems, such as planning. Indeed, many theorem provers (e.g., PTTP [49]) are built as backward reasoners and must have a query or *goal* in order to run.

One way to use MP for an analogous backward message-passing scheme is to assert $\neg Q$ in $\mathcal{A}_k$, choose a partition $\mathcal{A}_j$ that is most distant from $\mathcal{A}_k$ in G, and try to prove $\{\}$ in $\mathcal{A}_j$. If we wish to follow the spirit of backward-reasoning more closely, we can connect all the nodes of G, and run BREAK-CYCLES with the stipulation that it must produce a *chain* graph with $\mathcal{A}_k$ at one end. The resultant chain graph may then be used for query-driven backward message-passing, from $\mathcal{A}_k$. With resolution, the goal of a partition at a given time can be taken to be the disjunction of the negation of the messages it has received. Note that for the algorithm to be complete, each partition's reasoner must be complete for consequence finding.

2.2 Resolution-Based Inference

We now analyze the effect of forward message-passing (MP) on the computational efficiency of resolution-based inference, and identify some of the parameters of influence. Current measures for comparing automated deduction strategies are insufficient for our purposes. Proof length (e.g., [28]) is only marginally relevant. More relevant is comparing the sizes of search spaces of different strategies (e.g., [41]). These measures do not precisely address our needs, but we use them here, leaving better comparison for future work.

In a *resolution search space*, each node includes a set of clauses, and properties relevant to the utilized resolution strategy (e.g., clause parenthood information). Each arc is a resolution step allowed by the strategy. In contrast, in an *MP resolution search space* the nodes also include partition membership information. Further, each arc is a resolution step allowed by the utilized resolution strategy that satisfies either of: (1) the two axioms are in the same partition, or (2) one of the axioms is in partition $\mathcal{A}_j$, the second axiom is drawn from its communication language $l(i,j)$, and the query-based ordering allows the second axiom to be sent from $\mathcal{A}_i$ to $\mathcal{A}_j$. Legal sequence of resolutions correspond to paths in these spaces.

Proposition 2.4 *Let $\mathcal{A} = \bigcup_{i \leq n} \mathcal{A}_i$ be a partitioned theory. Any path in the MP resolution search space of $\{\mathcal{A}_i\}_{i \leq n}$ is also a path in the resolution search space of the unpartitioned theory $\mathcal{A}$.*

A similar proposition is true for linear resolution (each partition $\mathcal{A}_i$ uses linear resolution in which messages to $\mathcal{A}_i$ are treated as regular (non-input) sentences in $\mathcal{A}_i$).

From the point of view of proof length, it follows that the longest proof without using MP is as long or longer than the longest MP proof. Unfortunately, the shortest MP proof may be longer than the shortest possible proof without MP. This observation can be quantified most easily in the simple case of only two partitions $\mathcal{A}_1, \mathcal{A}_2$. The set of messages that need to be sent from $\mathcal{A}_1$ to $\mathcal{A}_2$ to prove Q is exactly the interpolant γ promised by Theorem 2.1 for $\alpha = A_1$, $\beta = \mathcal{A}_2 \Rightarrow Q$. The MP proof has to prove $\alpha \vdash \gamma$ and $\gamma \vdash \beta$. Carbone [9] showed that, if γ is a minimal interpolant,

then for many important cases the proof length of $\alpha \vdash \gamma$ together with the proof length of $\gamma \vdash \beta$ is in $O(k^2)$ (for sequent calculus with cuts), where k is the length of the minimal proof of $\alpha \vdash \beta$.

In general, the size of γ itself may be large. In fact, in the propositional case it is an open question whether or not the size of the smallest interpolant can be polynomially bounded by the size of the two formulae α, β. A positive answer to this question would imply an important consequence in complexity theory, namely that $NP \cap coNP \subseteq P/poly$ [7]. Nevertheless, there is a good upper bound on the length of the interpolation formula as a function of the length of the minimal proof [32] : If α, β share l symbols, and the resolution proof of $\alpha \vdash \beta$ is of length k, then there is an interpolant γ of length $min(kl^{O(1)}, 2^l)$.

Thus, we can guarantee a small interpolant, if we make sure the communication language is minimal. Unfortunately, we do not always have control over the communication language, as in the case of multiple KBs that have extensive overlap. In such cases, the communication language between KBs may be large, possibly resulting in a large interpolant. In Section 4 we provide an algorithm for partitioning theories that attempts to minimize the communication language between partitions.

3 Propositional Satisfiability

In this section we propose an algorithm for partition-based logical reasoning based on propositional satisfiability (SAT) search. We show that the complexity of computation is directly related to the size of the labels in the intersection graph.

3.1 A Partition-Based SAT Procedure

The algorithm we propose uses a SAT procedure as a subroutine and is back-track free. We describe the algorithm using database notation [51]. $\pi_{p_1,...,p_k}T$ is the *projection* operation on a relation T. It produces a relation that includes all the rows of T, but only the columns named $p_1, ..., p_k$ (suppressing duplicate rows). $S \bowtie R$ is the *natural join* operation on the relations S and R. It produces the cross product of S, R, selecting only those entries that are equal between identically named fields (checking $S.A = R.A$), and discarding those columns that are now duplicated (e.g., $R.A$ will be discarded).

The proposed algorithm shares some intuition with prime implicate generation (e.g., [36, 30]). Briefly, we first compute all the models of each of the partitions (akin to computing the implicates of each partition). We then use $\bowtie$ to combine the partition models into models for $\mathcal{A}$. The algorithm is presented in Figure 7.

PROCEDURE LINEAR-PART-SAT($\{\mathcal{A}_i\}_{i \leq n}$)
$\{\mathcal{A}_i\}_{i \leq n}$ a partitioning of the theory $\mathcal{A}$,

1. $G_0 \leftarrow$ the intersection graph of $\{\mathcal{A}_i\}_{i \leq n}$. $G \leftarrow BREAK\text{-}CYCLES(G_0)$.
2. For each $i \leq n$, let $L(i) = \bigcup_{(i,j) \in E} l(i,j)$.
3. For each $i \leq n$, for every truth assignment A to $L(i)$, perform SAT-search on $\mathcal{A}_i \cup A$, storing the result in a table $T_i(A)$.
4. Let $dist(i,j)$ $(i,j \in V)$ be the length of the shortest path between i, j in G. Let $i \prec j$ iff $dist(i,1) < dist(j,1)$ ($\prec$ is a strict partial order).
5. Iterate over $i \leq n$ in reverse $\prec$-order (the last i is 1). For each $j \leq n$ that satisfies $(i,j) \in E$ and $i \prec j$, perform:
   - $T_i \leftarrow T_i \bowtie (\pi_{L(i)} T_j)$ (*Join* T_i with those columns of T_j that correspond to $L(i)$). If $T_i = \emptyset$, return FALSE.
6. Return TRUE.

Figure 7: An algorithm for SAT of a partitioned propositional theory.

The iterated join that we perform takes time proportional to the size of the tables involved. We keep table sizes below $2^{|L(i)|}$ ($L(i)$ computed in step 2), by *projecting* every table before *joining* it with another. Soundness and completeness follow by an argument similar to that given for MP.

Theorem 3.1 (Soundness and Completeness) *Given a sound and complete SAT-search procedure, LINEAR-PART-SAT is sound and complete for SAT of partitioned propositional theories.*

PROOF See Appendix A.3.

3.2 Analyzing Satisfiability in LINEAR-PART-SAT

Let $\mathcal{A}$ be a partitioned propositional theory with n partitions. Let $m = |\mathcal{L}(\mathcal{A})|$, $L(i)$ the set of propositional symbols calculated in step 2 of LINEAR-PART-SAT, and $m_i = |\mathcal{L}(\mathcal{A}_i) \setminus L(i)|$ $(i \leq n)$. Let $a = |\mathcal{A}|$ and k be the length of each axiom.

Lemma 3.2 *The time taken by LINEAR-PART-SAT to compute SAT for $\mathcal{A}$ is*

$$Time(n, m, m_1, ..., m_n, a, k, |L(1)|, ..., |L(n)|) = O(a * k^2 + n^4 * m + \sum_{i=1}^{n} (2^{|L(i)|} * f_{SAT}(m_i))),$$

where f_{SAT} is the time to compute SAT. If the intersection graph G_0 is a tree, the second argument in the summation can be reduced from $n^4 * m$ to $n * m$.

PROOF See Appendix A.4.

Corollary 3.3 *Let $\mathcal{A}$ be a partitioned propositional theory with n partitions, m propositional symbols, and intersection graph $G = (V, E, l)$. Let $d = max_{v \in V} d(v)$, where $d(v)$ is the degree of node v, and let $l = max_{i,j \leq n} |l(i,j)|$. Assume $P \neq NP$. If intersection graph G of $\mathcal{A}$ is a tree and all the partitions $\mathcal{A}_i$ have the same number of propositional symbols, then the time taken by the LINEAR-PART-SAT procedure to compute SAT for $\mathcal{A}$ is*

$$Time(m, n, l, d) = O(n * 2^{d*l} * f_{SAT}(\frac{m}{n})).$$

For example, if we partition a given theory $\mathcal{A}$ into only two partitions ($n = 2$), sharing l propositional symbols, the algorithm will take time $O(2^l * f_{SAT}(\frac{m}{2}))$. Assuming $P \neq NP$, this is a significant improvement over a simple SAT procedure, for every l that is small enough ($l < \frac{\alpha m}{2}$, and $\alpha \leq 0.582$ [42, 13]).

4 Decomposing a Logical Theory

The algorithms presented in previous sections assumed a given partitioning. In this section we address the critical problem of automatically decomposing a set of propositional or FOL clauses into a partitioned theory. Guided by the results of previous sections, we propose guidelines for achieving a good partitioning, and present a greedy algorithm that decomposes a theory following these guidelines.

4.1 A Good Partitioning

Given a theory, we wish to find a partitioning that minimizes the formula derived in Lemma 3.2. To that end, assuming $P \neq NP$, we want to minimize the following parameters in roughly the following order. For all $i \leq n$:

1. $|L(i)|$ - the total number of symbols contained in all links to/from node i. If G_0 is already a tree, this is the number of symbols shared between the partition $\mathcal{A}_i$ and the rest of the theory $\mathcal{A} \setminus \mathcal{A}_i$.
2. m_i - the number of symbols in a partition, less those in the links, i.e., in $\mathcal{A}_i \setminus L(i)$. This number is mostly influenced by the size of the original partition $\mathcal{A}_i$, which in turn is influenced by the number of partitions of $\mathcal{A}$, namely, n. Having more partitions will cause m_i to become smaller.
3. n - the number of partitions.

Also, a simple analysis shows that given *fixed* values for l, d in Corollary 3.3, the maximal n that maintains l, d such that also $n \leq ln2 * \alpha * m$ ($\alpha = 0.582$ [42, 13]) yields an optimal bound for LINEAR-PART-SAT. In Section 2.2 we saw that the same parameters influence the number of derivations we can perform in MP: $|L(i)|$ influences the interpolant size and thus the proof length, and m_i influences the number of deductions/resolutions we can perform. Thus, we would like to minimize the number of symbols shared between partitions and the number of symbols in each partition less those in the links.

The question is, how often do we get large n (many partitions), small m_i's (small partitions) and small $|L(i)|$'s (weak interactions) in practice. We believe that in domains that deal with engineered physical systems, many of the domain axiomatizations have these structural properties. Indeed, design of engineering artifacts encourages modularization, with minimal interconnectivity (see [2, 34, 12]). More generally, we believe axiomatizers of large corpora of real-world knowledge tend to try to provide structured representations following some of these principles.

4.2 Vertex Min-Cut in the Graph of Symbols

To exploit the partitioning guidelines proposed in the previous subsection, we represent our theory $\mathcal{A}$ using a *symbols graph* that captures the features we wish to minimize. $G = (V, E)$ is a symbols graph for theory $\mathcal{A}$ such that each vertex $v \in V$ is a symbol in $\mathcal{L}(\mathcal{A})$, and there is an edge between two vertices if their associated symbols occur in the same axiom of $\mathcal{A}$, i.e., $E = \{(a, b) \mid \exists \alpha \in \mathcal{A}$ s.t. a, b appear in $\alpha\}$.

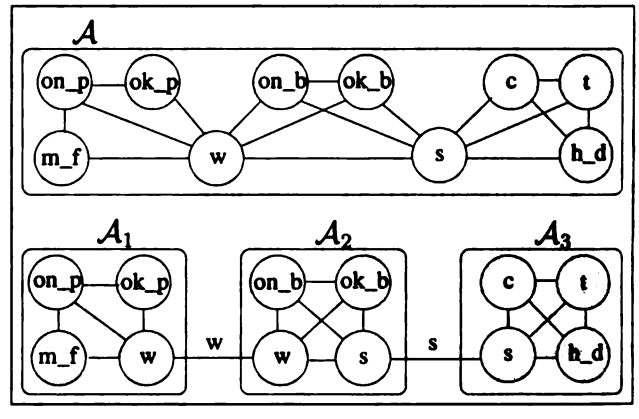


Figure 8: Decomposing $\mathcal{A}$'s symbols graph.

Figure 8 (top) illustrates the symbols graph of theory $\mathcal{A}$ from Figure 1 and the connected symbols graphs (bottom) of the individual partitions $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$. The symbols *ok_p*, *on_p*, *m_f*, *w*, *ok_b*, *on_b*, *s*, *c*, *t*, *h_d* are short for *ok_pump*, *on_pump*, *man_fill*, *water*, *ok_boiler*, *on_boiler*, *steam*, *coffee*, *teabag*, *hot_drink*, respectively. Notice that each axiom creates a clique among its constituent symbols. To minimize the number of symbols shared between partitions (i.e., $|L(i)|$), we must find parti-

tions whose symbols have minimal *vertex separators* in the symbols graph.

We briefly describe the notion of a vertex separator. Let $G = (V, E)$ be an undirected graph. A set S of vertices is called an (a, b) *vertex separator* if $\{a, b\} \subset V \setminus S$ and every path connecting a and b in G passes through at least one vertex contained in S. Thus, the vertices in S split the path from a to b. Let $N(a, b)$ be the least cardinality of an (a, b) vertex separator. The *connectivity* of the graph G is the minimal $N(a, b)$ for any $a, b \in V$ that are not connected by an edge.

Even [22] described algorithms for finding minimum vertex separators, building on work of Dantzig and Fulkerson. We briefly review one of these algorithms before we use it to decompose our theories. It is shown in Figure 9. The algorithm is given two vertices, a, b, and an undirected graph, G. In this algorithm, we transform G into $\bar{G}$ and run a max-flow algorithm on it (lines 1–3). The produced flow, f, has a throughput of $N(a, b)$. To extract a minimal separator, we produce a *layered network* (see [22] p.97) from $\bar{G}, f$ in line 5. The layered network includes a subset of the vertices of $\bar{G}$. The set of edges between this set of vertices and the rest of $\bar{G}$ corresponds to the separator.

PROCEDURE MIN-V-SEP-A-B($G = (V, E)$, a, b)

1. Construct a digraph $\bar{G}(\bar{V}, \bar{E})$ as follows. For every $v \in V$ put two vertices v', v'' in $\bar{V}$ with an edge $e_v = \overrightarrow{(v', v'')}$ (*internal edge*). For every edge $e = (u, v)$ in G, put two edges $e' = \overrightarrow{(u'', v')}$ and $e'' = \overrightarrow{(u', v'')}$ in $\bar{G}$ (*external edges*).
2. Define a network, with digraph $\bar{G}$, source a'', sink b' and unit capacities for all the edges.
3. Compute the *maximum flow* f in the network. $N(a, b)$ is the throughput of f.
4. Set the capacities of all the external edges in $\bar{G}$ to infinity.
5. Construct the *layered network* $\{V_i\}_{i \leq l}$ from $\bar{G}$ using f. Let $S = \bigcup_{i \leq l} V_i$.
6. Let $R = \{v \in V \mid v' \in S,\ v'' \notin S\}$. R is a minimum (a, b) vertex-separator in G.

Figure 9: An algorithm for finding a minimal separator between a and b in G.

Algorithms for finding maximal flow are abundant in the graph algorithms literature. Prominent algorithms for max-flow include the Simplex method, Ford and Fulkerson's [31], the push-relabel method of Goldberg and Tarjan [27] (time bound of $O(|V| * |E| * lg\frac{|V|^2}{|E|}$) and several implementations [11]), and Dinitz's algorithm [21]. When Dinitz's algorithm is used to solve the network problem, algorithm MIN-V-SEP-A-B is of time complexity $O(|V|^{\frac{1}{2}} * |E|)$ [22].

Finally, to compute the vertex connectivity of a graph and a minimum separator, without being given a pair (a, b), we check the connectivity of any c vertices (c being the connectivity of the graph) to all other vertices. When Dinitz's algorithm is used as above, this procedure takes time $O(c * |V|^{\frac{3}{2}} * |E|)$, where $c \geq 1$ is the connectivity of G. For the cases of $c = 0, 1$ there are well known linear time algorithms.

Figure 10 presents a greedy recursive algorithm that uses Even's algorithm to find *sets of vertices* that together separate a graph into partitions. The algorithm returns a set of symbols sets that determine the separate subgraphs. Different variants of the algorithm yield different structures

PROCEDURE SPLIT(G, M, l, a, b)
$G = (V, E)$ is an undirected graph. M is the limit on the number of symbols in a partition. l is the limit on the size of links between partitions. a, b are in V or are nil.

1. If $|V| < M$ then return the graph with the single symbol set V.
2. (a) If a and b are both nil, find a minimum vertex separator R in G. (b) Otherwise, if b is nil, find a minimum vertex separator R in G that does not include a. (c) Otherwise, find a minimum vertex separator R in G that separates a and b.
   If $R > l$ then return the graph with the single symbol set V.
3. Let G_1, G_2 be the two subgraphs of G separated by R, with R included in both subgraphs.
4. Create G_1', G_2' from G_1, G_2, respectively, by aggregating the vertices in R into a single vertex r, removing all self edges and connecting r with edges to all the vertices connected by edges to some vertices in R.
5. Set $V^1 = SPLIT(G_1', M, l, r, a)$ and $V^2 = SPLIT(G_2', M, l, r, b)$.
6. Replace r in V^1, V^2 by the members of R. Return V^1, V^2.

Figure 10: An algorithm for generating symbol sets that define partitions.

for the intersection graph of the resulting partitioning. As is, SPLIT returns sets of symbols that result in a chain of partitions. We obtain arbitrary *trees*, if we change step 2(c) to find a minimum separator that does not include a, b (not required to separate a, b). We obtain arbitrary *graphs*, if in addition we do not aggregate R into r in step 4.

Proposition 4.1
Procedure SPLIT takes time $O(|V|^{\frac{5}{2}} * |E|)$.

PROOF See Appendix A.5.

Finally, to partition a theory $\mathcal{A}$, create its symbols graph G

and run SPLIT(G, M, l, nil, nil). For each set of symbols returned, define a partition $\mathcal{A}_i$ that includes all the axioms of $\mathcal{A}$ in the language defined by the returned set of symbols.

We know of no easy way to find an optimal selection of l (the limit on the size of the links) and M (the limit on the number of symbols in a partition) without having prior knowledge of the dependency between the number (and size) of partitions and l. However, we can find out when a partitioning no longer helps computation (compared to the best time bound known for SAT procedures [42, 13]). Our time bound for the procedure is lower than $\Theta(2^{\alpha m})$ when $l \leq \frac{\alpha m - \alpha m_i - lgn}{d}$ ($i = argmax_j m_j$). In particular, if $l > \frac{m}{2}$, a standard deterministic SAT procedure will be better. Hence, l and M are perhaps best determined experimentally.

In contrast to the SPLIT procedure, there are other approaches that can be taken. One approach that we have also implemented is a normalized cut algorithm [44], using the dual graph of the theory. The dual graph represents each axiom, rather than each symbol, as a node in the graph to be split. The advantage of this approach is that it preserves the distinction of an axiom. Also, since the min-cut algorithm is normalized, it helps preclude the creation of small isolated partitions by both maximize the similarity within partitions and minimizing the similarity between partitions. Finally, our reasoning algorithms and our computational analysis suggested a syntactic approach to decomposition. Semantic approaches are also possible along lines similar to [10] (Ch. on semantic resolution) or [48]. Such decomposition approaches may require different reasoning algorithms to be useful.

5 Related Work

There has been little work on the specific problem of exploiting structure in theorem proving and SAT search in the manner we propose, and little work on automatically partitioning logical theories for this purpose. Nevertheless, there are many areas of related work.

Work on formalizing and reasoning with *context* (e.g., [37, 1]) can be related to partition-based logical reasoning by viewing the contextual theories as interacting sets of theories. Unfortunately, to introduce explicit contexts, a language that is more expressive than FOL is needed. Consequently, a number of researchers have focused on context for propositional logic, while much of the reasoning work has focused on proof checking (e.g., GETFOL [26]), There have been few reported successes with automated reasoning; [6] presents one example.

Many AI researchers have exploited some type of structure to improve the efficiency of reasoning (e.g., Bayes Nets [40], Markov decision processes [8], CSPs [19, 18], and model-based diagnosis [17]). There is also a vast literature in both clustering and decomposition techniques. Most relevant to our work, are CSP decomposition techniques that look for a *separation vertex* [18]. Also related is cutset conditioning [40]. In contrast, our work focuses on the possibility of using vertex separator *sets* as a generalization of the separation-vertex concept. Partly due to this generality, our work is the first to address the problem of defining guidelines and parameters for good decompositions of sets of axioms for the purpose of logical reasoning.

Decomposition has not been exploited in theorem proving until recently (see [4, 5]). We believe that part of the reason for this lack of interest has been that theorem proving has focused on mathematical domains that do not necessarily have structure that supports decomposition. Work on theorem proving has focused on decomposition for parallel implementations [5, 14, 50] and has followed decomposition methods guided by lookahead and subgoals, neglecting the types of structural properties we used here. Another related line of work focuses on combining logical systems (e.g., [38, 45, 3]). Contrasted with this work, we focus on interactions between theories with overlapping signatures, the efficiency of reasoning, and automatic decomposition.

Decomposition for propositional SAT has followed different tracks. Some work focused on heuristics for clause weighting or symbol ordering (e.g., [43, 20]). [39] suggested a decomposition procedure that represents the theory as a hypergraph of clauses and divides the propositional theory into two partitions (heuristically minimizing the number of hyperedges) modifying ideas described in [25]. [14] developed an algorithm that partitions a propositional theory into connected components. Both [14, 39] performed experiments that demonstrated a decrease in the time required to prove test sets of axioms unsatisfiable.

6 Conclusions

We have shown that decomposing theories into partitions and reasoning over those partitions has computational advantages for theorem provers and SAT solvers. Theorem proving strategies, such as resolution, can use such decompositions to constrain search. Partition-based reasoning will improve the efficiency of propositional SAT solvers if the theory is decomposable into partitions that share only small numbers of symbols. We have provided sound and complete algorithms for reasoning with partitions of related logical axioms, both in propositional and FOL. Further, we analyzed the effect of partition-based logical reasoning on resolution-based inference, both with respect to proof search space size, and with respect to the length of a proof. We also analyzed the performance of our SAT algorithm and showed that it takes time proportional to SAT solutions on individual partitions and an exponent in the size of the

links between partitions. Both algorithms can gain further time efficiency through parallel processing.

Guided by the analysis of our SAT algorithm, we suggested guidelines for achieving a good partitioning and proposed an algorithm for the automatic decomposition of theories that tries to minimize identified parameters. This algorithm generalizes previous algorithms used to decompose CSPs by finding single-vertex separators.

Our work was motivated in part by the problem of reasoning with large multiple KBs that have overlap in content. The results in this paper address some of the theoretical principles that underly such partition-based reasoning. In future work, we plan an experimental analysis of Stanford KSL's and SRI's KBs to analyze structure in these KBs, to test the effectiveness of our automatic partitioning algorithms, and to investigate the effectiveness of proposed partition-based reasoning algorithms. We are also involved in further theoretical investigation.

A Proofs

A.1 Proof of Theorem 2.2

First, notice that soundness is immediate because the only rules used in deriving consequences are those used in our chosen consequence-finding procedure (of which rules are sound). In all that follows, we assume $\mathcal{A}$ is finite. The infinite case follows by the compactness of FOL.

Lemma A.1 *Let $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ be a partitioned theory. Let $\varphi \in \mathcal{L}(\mathcal{A}_2)$. If $\mathcal{A} \vdash \varphi$, then $\mathcal{A}_2 \vdash \varphi$ or there is a sentence $\psi \in \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$ such that $\mathcal{A}_1 \vdash \psi$ and $\mathcal{A}_2 \vdash \psi \Rightarrow \varphi$.*

Proof of Lemma A.1. We use Craig's interpolation theorem (Theorem 2.1), taking $\alpha = \mathcal{A}_1$ and $\beta = \mathcal{A}_2 \Rightarrow \varphi$. Since $\alpha \vdash \beta$ (by the deduction theorem for FOL), there is a formula $\psi \in \mathcal{L}(\alpha) \cap \mathcal{L}(\beta)$ such that $\alpha \vdash \psi$ and $\psi \vdash \beta$. By the deduction theorem for FOL, we get that $\mathcal{A}_1 \vdash \psi$ and $\psi \wedge \mathcal{A}_2 \vdash \varphi$. Since $\psi \in \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$ by the way we constructed α, β, we are done. ■

Definition A.2 *For a partitioning $\mathcal{A} = \bigcup_{i \le n} \mathcal{A}_i$, we say that a tree $G = (V, E, l)$ is* properly labeled, *if for all $(i,j) \in E$ and $\mathcal{B}_1, \mathcal{B}_2$ the two subtheories of $\mathcal{A}$ on the two sides of the edge (i,j) in G, it is true that $\mathcal{L}(l(i,j)) \supseteq \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$.*

We will show that all intersection graphs are properly labeled. First, the following lemma provides the main argument behind all of the completeness proofs in this paper.

Lemma A.3 *Let $\mathcal{A} = \bigcup_{i \le n} \mathcal{A}_i$ be a partitioned theory and assume that the graph G is a tree that is properly labeled for the partitioning $\{\mathcal{A}_i\}_{i \le n}$. Let $k \le n$ and let $Q \in \mathcal{L}(\mathcal{A}_k \cup \bigcup_{(k,i) \in E} l(k,i))$ be a sentence. If $\mathcal{A} \models Q$, then MP will output YES.*

Proof of Lemma A.3. We prove the lemma by induction on the number of partitions in the logical theory. For $|V| = 1$ (a single partition), $\mathcal{A} = \mathcal{A}_1$ and the proof is immediate. Assume that we proved the lemma for $|V| \le n-1$ and we prove the lemma for $|V| = n$.

In G, k has c neighbors, $i_1, ..., i_c$. (k, i_1) separates two parts of the tree G: G_1 (includes i_1) and G_2 (includes k). Let $\mathcal{B}_1, \mathcal{B}_2$ be the subtheories of $\mathcal{A}$ that correspond to G_1, G_2, respectively.

Notice that $Q \in \mathcal{L}(\mathcal{B}_2)$. By Lemmma A.1, either $\mathcal{B}_2 \vdash Q$ or there is $\psi \in \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$ such that $\mathcal{B}_1 \vdash \psi$ and $\mathcal{B}_2 \vdash \psi \Rightarrow Q$. If $\mathcal{B}_2 \vdash Q$, then we are done, by the induction hypothesis applied to the partitioning $\{\mathcal{A}_i \mid i \in V_2\}$ (V_2 includes the vertices of G_2) and G_2 (notice that $\prec'$ used for G_2, Q agrees with $\prec$ used for G).

Otherwise, let ψ be a sentence as above. $\bigcup_{(i_1,j) \in E, j \neq k} l(i_1, j) \supseteq \mathcal{L}(\mathcal{B}_2 \cup \mathcal{A}_{i_1}) \cap \mathcal{L}(\mathcal{B}_1 \setminus \mathcal{A}_{i_1})$ because the set of edges (i_1, j) separates two subgraphs corresponding to the theories $\mathcal{B}_1 \setminus \mathcal{A}_{i_1}$ and $\mathcal{B}_2 \cup \mathcal{A}_{i_1}$, and G is properly labeled for our partitioning. Thus, since $\psi \in \mathcal{L}(\mathcal{B}_1)$ we get that $\psi \in \mathcal{L}(\mathcal{A}_{i_1} \cup \bigcup_{(i_1,j) \in E, j \neq k} l(i_1, j))$. By the induction hypothesis for $G_1, \mathcal{B}_1$, at some point ψ will be proved in $\mathcal{A}_i$ (after some formulae were sent to it from the other partitions in G_1, B_1).

At this point, our algorithm will send ψ to $\mathcal{A}_k$ because $\psi \in \mathcal{L}(l(k, i_1))$ because G is properly labeled for $\mathcal{A}, G$. Since $\mathcal{B}_2 \vdash \psi \Rightarrow Q$, then by the induction hypothesis applied to $G_2, \mathcal{B}_2$ ($\psi \Rightarrow Q \in \mathcal{L}(\mathcal{A}_k \cup_{k,i) \in E} l(k,i))$ at some point $\psi \Rightarrow Q$ will be proved in $\mathcal{A}_k$ (after some message passing). Thus, at some point $\mathcal{A}_k$ will prove Q. ■

Proof of Theorem 2.2. All we are left to prove is that the intersection graph G is properly labeled. But if G is the intersection graph of the partitioning $\{\mathcal{A}_i\}_{i \le n}$ then $l(i,j) = \mathcal{L}(\mathcal{A}_i) \cap \mathcal{L}(\mathcal{A}_j)$. If for $(i,j) \in E$ $\mathcal{L}(l(i,j)) \not\supseteq \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$, with $\mathcal{B}_1, \mathcal{B}_2$ the theories on the two sides of (i,j) in the tree G, then there are $\mathcal{A}_x, \mathcal{A}_y$ in $\mathcal{B}_1, \mathcal{B}_2$, respectively, such that $(x,y) \in E$ and $x \neq i$ or $y \neq j$. Since G is connected (it is a single tree), this means there is a cycle in G, contradicting G being a tree. Thus $\mathcal{L}(l(i,j)) \supseteq \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$ and G is properly labeled. The proof follows from Lemma A.3. ■

A.2 Proof of Theorem 2.3

Soundness is immediate, using the same argument as for Theorem 2.2. For completeness, first notice that the graph output by BREAK-CYCLES is always a tree, because

BREAK-CYCLES will not terminate if there is still a cycle in G. Now, we need the following lemma.

Lemma A.4 *Let $G' = (V, E', l')$ be a tree resulting from applying BREAK-CYCLES to $G = (V, E, l)$ and $\{\mathcal{A}_i\}_{i \leq n}$. Then G' is properly labeled for this partitioning.*

Proof of Lemma A.4. Assume there is a symbol p in $\mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$ that is not in $l(i,j)$, and let $\mathcal{A}_x, \mathcal{A}_y$ be partitions on the two sides of (i,j) that include sentences with the symbol p. We will prove that throughout the run of the BREAK-CYCLES algorithm there is always a path in G' (we start with $G' = G$) between $\mathcal{A}_x, \mathcal{A}_y$ that has p showing on all the edge labels. Call such a path a *good path*.

Obviously we have a good path in G, because we have $(x,y) \in E$ and $p \in l(x,y)$ (because G is the intersection graph of $\mathcal{A}_1, ..., \mathcal{A}_n$). Let us stop the algorithm at the first step in which G' does not have a good path (assuming there is no such path, or otherwise we are done). In the last step we must have removed an arc (a,b) (which was on a good path) to cause G' to not have a good path. Since $p \in l(a,b)$ and (a,b) is in a cycle $\langle (b,a_1), (a_1,a_2), ..., (a_c,a), (a,b) \rangle$ (this is the only reason we removed (a,b)), we added $l(a,b)$ to the labels of the rest of this cycle. In particular, now the labels of $(b,a_1), (a_1,a_2), ..., (a_c,a)$ include p. Replacing (a,b) in the previous good path by this sequence, we find a path in the new G' that satisfies our required property. This is a contradiction to having assumed that there is no such path at this step. Thus, there is no such p as mentioned above and $\mathcal{L}(l(i,j)) \supseteq \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$. ■

Proof of Theorem 2.3. The proof of Theorem 2.3 follows immediately from Lemma A.3 and Lemma A.4. ■

A.3 Proof of Theorem 3.1

Proof of Theorem 3.1. For each partition $\mathcal{A}_i, i \leq n$, lines 1 – 3 perform the equivalent of finding all the models of $\mathcal{A}_i$ and storing their truth assignments to the symbols of $\mathcal{L}(L(i))$ in T_i. ($L(i)$ specifies the columns, thus each row corresponds to a truth assignment.) This is equivalent to finding the implicates of the theory $\mathcal{A}_i$ in the sublanguage $\mathcal{L}(L(i))$. Thus, if A_i is the DNF of the set of implicates $(\alpha_1(p_{j_1}, ..., p_{j_{l_1}}) \vee ... \vee \alpha_{a_i}(p_{j_1}, ..., p_{j_{l_{a_i}}}))$, then T_i initially includes the set of models of A_i in the sublanguage $\mathcal{L}(L(i))$, namely, $[\![A_i]\!]_{\mathcal{L}(L(i))}$.

The *natural join* operation ($\bowtie$) then creates all the consistent combinations of models from $[\![A_i]\!]_{\mathcal{L}(L(i))}$ and $[\![A_j]\!]_{\mathcal{L}(L(j))}$. This set of consistent combinations is the set of models of $A_i \cup A_j$. Thus, $T_i \bowtie T_j \equiv [\![(A_i \cup A_j)]\!]_{\mathcal{L}(L(i) \cup L(j))}$.

Finally, the *projection* operation restricts the models to the sublanguage $\mathcal{L}(L(i))$, getting rid of duplicates in the sublanguage. This is equivalent to finding all the implicates of $A_i \wedge A_j$ in the sublanguage $\mathcal{L}(L(i))$. Thus, $\pi_{L(i)}(T_i \bowtie T_j) \equiv [\![\{\varphi \in \mathcal{L}(L(i)) \mid A_i \cup A_j \models \varphi\}]\!]_{\mathcal{L}(L(i))}$.

To see that the algorithm is sound and complete, notice that the it does the analogous operations to our forward message-passing algorithm MP (Figure 3). We break the cycles in G_0 (creating G) and perform forward reasoning as in MP, using the set of implicates instead of online reasoning in each partition: instruction 3 in MP is our projection ("$\mathcal{A}_i \models \varphi$ and $\varphi \in \mathcal{L}(l(i,j))$") and then join ("add φ to the set of axioms of $\mathcal{A}_j$"). Since $T_i \bowtie T_j \equiv [\![(A_i \cup A_j)]\!]_{\mathcal{L}(L(i) \cup L(j))}$, joining corresponds to sending all the messages together. Since $\pi_{L(i)}(T_i \bowtie T_j) \equiv [\![\{\varphi \in \mathcal{L}(L(i)) \mid A_i \cup A_j \models \varphi\}]\!]_{\mathcal{L}(L(i))}$, projection corresponds to sending only those sentences that are allowed by the labels.

By Theorem 2.3, LINEAR-PART-SAT is sound and complete for satisfiability of $\mathcal{A}$. ■

A.4 Proof of Lemma 3.2

Proof of Lemma 3.2. Let $\mathcal{A}$ be a partitioned propositional theory with n partitions. Let m be the total number of propositional symbols in $\mathcal{A}$, $L(i)$ the set of propositional symbols calculated in step 2 of LINEAR-PART-SAT, and m_i the number of propositional symbols mentioned in $\mathcal{A}_i \setminus L(i)$ ($i \leq n$). Let us examine procedure LINEAR-PART-SAT (Figure 7) step by step, computing the time needed for computation.

Computing the intersection graph takes $O(a * k^2)$ time, where k is the number of propositional symbol in each axiom (for 3SAT, that is 3), because we check and add k^2 edges to G_0 for each axiom.

BREAK-CYCLES' loop starts by finding a minimal-length cycle, which takes time $O(n)$ (BFS traversal of n vertices). Finding the optimal a in line 2 takes time $O((c * m) * c)$, where c is the length of the cycle found (union of two labels takes at most $O(m)$ time). Finally, since a tree always satisfies $|E| = |V| - 1$, breaking all the cycles will require us to remove $|E| - (|V| - 1)$ edges. Thus, the loop will run $|E| - (|V| - 1)$ times (assuming the graph G_0 is connected). An upper bound on this algorithm's performance is then $O(n^2 * (n^2 * m)) = O(n^4 * m)$ (because $c \leq n$ and and $|E| \leq |V|^2 = n^2$).

Step 2 of LINEAR-PART-SAT takes time $O(n * m)$, since there are a total of $n - 1$ edges in the graph G (G is a tree with n vertices) and every label is of length at most m.

Checking the truth assignments in step 3 takes time $\sum_{i=1}^{n} 2^{|L(i)|}$ per satisfiability check of $\mathcal{A}_i \cup A$, because there are $2^{|L(i)|}$ truth assignments for each $i \leq n$. Since

$\mathcal{A}_i \cup A$ has only m_i free propositional variables, (A is an assignment of truth values to $|L(i)|$ variables), $\mathcal{A}_i \cup A$ is reducible (in time $O(|A|)$) to a theory of smaller size with only m_i propositional variables. If the time needed for a satisfiability check of a theory with m variables is $O(f_{SAT}(m))$, then the time for step 3 is

$$O(\sum_{i=1}^{n}(2^{|L(i)|} * f_{SAT}(m_i)))$$

Finding the relation $\prec$ takes $O(n)$ as it is easily generated by a BFS through the tree.

Instruction 5 performs a *projection* and *join*, which takes time $O(2^{|L(i)|})$ (the maximal size of the table). Since the number of iterations over $i \leq n$ and j being a child of i is $n-1$ (there are only $n-1$ edges), we get that the total time for this step is $O(\sum_{i=1}^{n} 2^{|L(i)|})$.

Summing up, the worst-case time used by the algorithm is

$$\begin{aligned} &Time(n, m, m_1, ..., m_n, a, k, L(1), ..., L(n)) = \\ &\quad O(a * k^2 + n^4 * m + n * m + \\ &\quad \sum_{i=1}^{n}(2^{|L(i)|} * f_{SAT}(m_i)) + n + \sum_{i=1}^{n} 2^{|L(i)|}) = \\ &\quad O(a * k^2 + n^4 * m + \sum_{i=1}^{n}(2^{|L(i)|} * f_{SAT}(m_i))). \end{aligned}$$

We can reduce the second argument (in the last formula) from $n^4 * m$ to $n * m$, if the intersection graph G_0 is already a tree. ■

A.5 Proof of Proposition 4.1

Proof of Proposition 4.1. Finding a minimum vertex separator R in G takes time $O(c * |V|^{\frac{3}{2}} * |E|)$. Finding a minimum separator that does not include s is equivalent to having s be the only source with which we check connectivity (in Even's algorithm). Thus, this can be done in time $O(|V|^{\frac{3}{2}} * |E|)$. Finding a minimum separator that separates s from t takes time $O(|V|^{\frac{1}{2}} * |E|)$. In the worst case, each time we look for a minimum s-separator ($t = nil$), we get a very small partition, and a very large one. Thus, we can apply this procedure $O(|V|)$ times. Summing up the time taken for each application of the procedure yields $O(|V| * |V|^{\frac{3}{2}} * |E| + c * |V|^{\frac{3}{2}} * |E|) = O(|V|^{\frac{5}{2}} * |E|)$. ■

Acknowledgements

We wish to thank Rada Chirkova and Tom Costello for helpful comments on the contents of this paper. We would also like to thank Mike Genesereth, Nils Nilsson and John McCarthy for many interesting discussions on more general topics related to this work. Finally, we thank the KR2000 reviewers for their helpful reviews. This research was supported in part by DARPA grant N66001-97-C-8554-P00004 and by AFOSR grant AF F49620-97-1-0207.

References

[1] V. Akman and M. Surav. Steps toward formalizing context. *AI Magazine*, 17(3):55–72, 1996.

[2] E. Amir. Object-Oriented First-Order Logic. *Linköping Electronic Articles in Computer and Information Science (http://www.ida.liu.se/ext/etai)*, 4, 1999. Under review in the RAC publication area. A preliminary version appeared in NRAC'99, an IJCAI-99 workshop.

[3] F. Baader and K. U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. In *11th Intl. conf. on automated deduction*, volume 607 of *LNAI*, pages 50–65. Springer-Verlag, 1992.

[4] M. P. Bonacina. A taxonomy of theorem-proving strategies. In *Artificial Intelligence Today – Recent Trends and Developments*, volume 1600 of *LNAI*, pages 43–84. Springer, 1999.

[5] M. P Bonacina and J. Hsiang. Parallelization of deduction strategies: an analytical study. *Journal of Automated Reasoning*, 13:1–33, 1994.

[6] P.E. Bonzon. A reflective proof system for reasoning in contexts. In *Proc. Nat'l Conf. on Artificial Intelligence (AAAI '97)*, pages 398–403, 1997.

[7] R. Boppana and M. Sipser. The complexity of finite functions. In *Handbook of Theoretical Computer Science*, volume 1. Elsevier and MIT Press, 1990.

[8] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. In *14th Intl. Joint Conf. on Artificial Intelligence (IJCAI '95)*, pages 1104–1111, 1995.

[9] A. Carbone. Interpolants, cut elimination and flow graphs for the propositional calculus. *Annals of Pure and Applied Logic*, 83(3):249–299, 1997.

[10] Chin-Liang Chang and R. Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.

[11] B. V. Chekassky and A. V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19(4):390–410, 1997.

[12] P. Cohen, R. Schrag, E. Jones, A. Pease, A. Lin, B. Starr, D. Gunning, and M. Burke. The DARPA high-performance knowledge bases project. *AI Magazine*, 19(4):25–49, 1998.

[13] S. A. Cook and D. G. Mitchell. Finding hard instances of the satisfiability problem: a survey. In *Dimacs Series in Discrete Mathamatics and Theoretical Computer Science*, volume 35. AMS, 1997.

[14] R. Cowen and K. Wyatt. BREAKUP: A preprocessing algorithm for satisfiability testing of CNF formulas. *Notre Dame J. of Formal Logic*, 34(4):602–606, 1993.

[15] W. Craig. Linear reasoning. a new form of the herbrand-gentzen theorem. *J. of Symbolic Logic*, 22:250–268, 1957.

[16] W. Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *J. of Symbolic Logic*, 22:269–285, 1957.

[17] A. Darwiche. Model-based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research*, 8:165–222, 1998.

[18] R. Dechter. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41(3):273–312, 1990.

[19] R. Dechter and J. Pearl. Tree Clustering Schemes for Constraint Processing. In *Natl' Conf. on Artificial Intelligence (AAAI '88)*, 1988.

[20] R. Dechter and I. Rish. Directional resolution: The davis-putnam procedure, revisited. In *Intl. Conf. on Knowledge Representation and Reasoning (KR '94)*, pages 134–145. Morgan Kaufmann, 1994.

[21] E. A. Dinitz. Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Mathematics Doklady*, 11:1277–1280, 1970.

[22] S. Even. *Graph Algorithms*. Computer Science Press, 1979.

[23] R. Fikes and A. Farquhar. Large-scale repositories of highly expressive reusable knowledge. *IEEE Intelligent Systems*, 14(2), 1999.

[24] J. J. Finger. Exploiting constraints in design synthesis. Technical Report STAN-CS-88-1204, Department of Computer Science, Stanford University, Stanford, CA, 1987.

[25] G. Gallo and G. Urbani. Algorithms for testing the satisfiability of propositional formulae. *J. of Logic Programming*, 7:45–61, 1989.

[26] E. Giunchiglia and P. Traverso. A multi-context architecture for formalizing complex reasoning. *International Journal of Intelligent Systems*, 10:501–539, 1995. Also, IRST Tech. Report #9307-26.

[27] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *J. of the ACM*, 35(4):921–940, 1988.

[28] A. Haken. The intractability of resolution. *theoretical computer science*, 39:297–308, 1985.

[29] G. Huang. Constructing Craig interpolation formulas. In *Conference on computing and combinatorics*, pages 181–190, 1995.

[30] K. Inoue. Linear resolution for consequence finding. *Artificial Intelligence*, 56(2-3):301–353, 1992.

[31] L. R. Ford Jr. and D. R. Fulkerson. *Flows in networks*. Princeton University Press, 1962.

[32] J. Krajiček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *J. of Symbolic Logic*, 62(2):457–486, 1997.

[33] R. C. Lee. *A Completeness Theorem and a Computer Program for Finding Theorems Derivable from Given Axioms*. PhD thesis, University of California, Berkeley, 1967.

[34] D. B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[35] R. C. Lyndon. An interpolation theorem in the predicate calculus. *Pacific J. of Mathematics*, 9(1):129–142, 1959.

[36] P. Marquis. Knowledge compilation using theory prime implicates. In *14th Intl. Joint Conf. on Artificial Intelligence (IJCAI '95)*, pages 837–843, 1995.

[37] J. McCarthy and S. Buvač. Formalizing Context (Expanded Notes). In A. Aliseda, R.J. van Glabbeek, and D. Westerståhl, editors, *Computing Natural Language*, volume 81 of *CSLI Lecture Notes*, pages 13–50. Center for the Study of Language and Information, Stanford U., 1998.

[38] G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM Trans. on Programming Languages and Systems*, 1(2):245–257, 1979.

[39] T. J. Park and A. Van Gelder. Partitioning methods for satisfiability testing on large formulas. In *Proc. Intl. Conf. on Automated Deduction (CADE-13)*, pages 748–762. Springer-Verlag, 1996.

[40] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[41] D. A. Plaisted. The search efficiency of theorem proving strategies. In *Proc. Intl. Conf. on Automated Deduction (CADE-12)*, pages 57–71, 1994.

[42] I. Schiermeyer. Pure literal look ahead: an $O(1,497^n)$ 3-satisfiability algorithm (extended abstract). Technical report, University of Köln, 1996. Workshop on the Satisfiability Problem, Siena April 29-May 3.

[43] B. Selman and H. Kautz. Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In *13th Intl. Joint Conf. on Artificial Intelligence (IJCAI '93)*, 1993.

[44] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proceedings of IEEE CVPR*, pages 731–737, 1997.

[45] R. E. Shostak. Deciding combinations of theories. *J. of the ACM*, 31:1–12, 1984.

[46] J. R. Slagle. Interpolation theorems for resolution in lower predicate calculus. *J. of the ACM*, 17(3):535–542, July 1970.

[47] J. R. Slagle, C.-L. Chang, and R. C. T. Lee. Completeness theorems for semantic resolution in consequence-finding. In *1st Intl. Joint Conf. on Artificial Intelligence (IJCAI '69)*, pages 281–285, 1969.

[48] J. Slaney and T.J. Surendonk. Combining finite model generation with theorem proving: Problems and prospects. In *Frontiers of Combining Systems: Proceedings of the 1st International Workshop, Munich (Germany)*, Applied Logic, pages 141–156. Kluwer, 1996.

[49] M. E. Stickel. A Prolog Technology Theorem Prover: a new exposition and implementation in Prolog. *Theoretical Computer Science*, 104:109–128, 1992.

[50] C. B. Suttner. SPTHEO. *Journal of Automated Reasoning*, 18:253–258, 1997.

[51] J. D. Ullman. *Principles of Database and knowledge-base systems*, volume 1. Computer Science Press, 1988.

Significant Inferences: Preliminary Report

Philippe Besnard
IRIT–CNRS
118 route de Narbonne
F–31062 Toulouse Cedex
besnard@irit.fr

Torsten Schaub
Institut für Informatik
Universität Potsdam
Postfach 60 15 53, D–14415 Potsdam
torsten@cs.uni-potsdam.de

Abstract

We explore the possibility of a logic where a conclusion substantially improves over its premise(s): Specifically, we intend to rule out inference steps such that the premise conveys more information, in a simpler form, than the conclusion does.

In fact, most reasoning formalisms, among them classical logic, come with means for generating disjunctive or conditional information in a fairly arbitrary way.

The basic principle for drawing disjunctive information is disjunctive weakening, which allows for deriving $\varphi \vee \psi$ from φ (for any ψ). Thus, given that "Nancy is married to Ron", disjunctive weakening makes us infer that "Nancy is married to Ron or Monica is married to Bill". Although the latter propositions may still be seemingly related, one should not forget (1) that any arbitrary proposition can serve as the additional disjunct, eg. "Nancy is married to Ron or the Queen of England is bald", and (2) that this can be iterated so that real information is buried in the generated disjunction among irrelevant propositions. What is the point in inferring such disjunctive formulas?

Similar phenomena can be traced back to conditionalization, which allows for deriving $\psi \rightarrow \varphi$ from φ (for any ψ).

As a result, we propose a natural deduction system along the intuitions sketched above.

1 MOTIVATION

Perhaps the most salient feature of reasoning is to make explicit what is only implicit. Accordingly, the less obvious a (correct) conclusion is, the more valuable is any reasoning by which that conclusion is drawn. In a sense, some conclusions may then not be worth inferring. Such a point of view is latent in relevance logics [Anderson and Belnap,1975] because they reject certain conclusions that weaken some premise in a peculiar way, as happens with arbitrary conditionalization for instance, i.e., $p \vdash q \rightarrow p$ (where p and q are propositional symbols). In this respect, relevance logics pave the way to a logic where only conclusions worth expressing are drawn. However, relevance logics stop short of fully achieving this idea. This can be viewed from relevantly valid schemes such as $p \vdash p \vee q$. Indeed, one may regard $p \vee q$ as a dubious weakening of p, as much as $q \rightarrow p$ is. In fact, we regard the scheme $p \vdash p \vee q$ as drawing a conclusion which fails to be significant *in view of* its premise. What do we mean by significant here? We mean that the conclusion is not worth stating when its premise is stated: Once it is clear that "the winning ticket is number 36", there is no point in making the inquiry whether "the winning ticket is number 17 or 36". Much as it would make no sense to investigate iteratively about the fact that "the winning ticket is number n_1 or... or n_k or 36" for k increasing. It does not even matter that any n_i be 36 as well: What is the point of stating that "the winning ticket is number 36 or the winning ticket is number 36" when it has already been stated that "the winning ticket is number 36"? That is, even the restricted form $p \vdash p \vee p$ is a particular form of dubious weakening.

Hence, we want to explore the possibility of a logic where a conclusion substantially improves over its premise(s): Specifically, we intend to rule out inference steps such that the premise conveys more infor-

mation, in a simpler form, than the conclusion does.

Indeed, we are not aiming at ignoring a conclusion that is less informative than its premises, provided that the conclusion has a simpler form. For instance,

$$p \vee q, p \rightarrow \neg r, s \leftrightarrow r, \neg p \rightarrow (q \rightarrow \neg s) \vdash \neg s$$

is an inference whose conclusion $\neg s$ does not exhaust all the information about p, q, r, s that the premises provide but is much easier to grasp.

2 TOWARDS SIGNIFICANT INFERENCE

In contrast to a proof system such as resolution [Robinson,1965] that has a single inference rule (ignoring factorization), natural deduction [Gentzen,1935] is traditionally a suitable framework for analyzing inferences. So let us consider the matter of *conditionalization* and *disjunctive weakening* from the perspective of natural deduction.

Conditionalization consists of turning an inference of ψ from some premises including φ into a proof of $\varphi \rightarrow \psi$. In extensional logics, the notion of being a premise is fairly liberal so that it is not required that φ actually serves for deriving ψ. Intensional logics such as relevance logics insist on φ being actually used to infer ψ: Relevance is then a necessary condition for $\varphi \rightarrow \psi$ to be derived. We adopt the same criterion because it matches also our idea of significance: on the one hand, $\varphi \rightarrow \psi$ is certainly significant whenever ψ is derived by means of φ, and, on the other hand, if ψ is derived independently of φ, then $\varphi \rightarrow \psi$ is both less simple and less insightful than ψ itself. In natural deduction, generalizing this leads us to requiring that the so-called "auxiliary assumptions" have to take part in the derivation of the conclusion that is declared to rely on them.

Disjunctive weakening consists of concluding $\varphi \vee \psi$ from φ (or similarly from ψ). We have argued above that this is never justified on its own, although it is sometimes useful as a device for inferring intermediate conclusions, as needed for reasoning by cases (whose principle is that, given $\varphi \vee \psi$, if χ is concluded from φ and if χ is concluded from ψ, then χ is inferred). So, weakening is needed for combining the conclusions obtained in each case:

$$p \vee q, p \rightarrow r, q \rightarrow s \vdash r \vee s \ .$$

In fact, we permit disjunctive weakening only for deriving conclusions drawn by reasoning by cases (see below).

Technically, banning disjunctive weakening while restricting conditionalization as just indicated could simply yield a subsystem of a relevance logic. However, we ban disjunctive weakening according to our intuitions about significant conclusions and these intuitions are different from those behind relevance logics, in particular, they make us depart from the relevance view against disjunctive syllogism (viz. $\varphi \vee \psi, \neg\varphi \vdash \psi$).

A well-taken objection by the relevantists is that if $\varphi \vee \psi$ holds because φ does then applying disjunctive syllogism is flawed: There is a contradiction between φ and $\neg\varphi$ but ψ is irrelevant in the matter. We agree that disjunctive syllogism is inappropriate in such a case but we contend that it is a valuable pattern when no contradiction is involved. Indeed, our intuitions about $\varphi \vee \psi$ are that the disjunction is really an alternative between φ and ψ so that it actually is about φ *as well as* about ψ. Accordingly, it is not about just φ or about just ψ and this warrants that if either case is denied then the other must hold.

As appears from the preceding discussion, significant inference is paraconsistent (i.e., the so-called *ex falso* $\varphi, \neg\varphi \vdash \psi$ does not hold). The intuitive notion of a significant conclusion appeals for paraconsistency in at least two different ways. One is that nothing is more informative or substantially simpler than $\varphi \wedge \neg\varphi$. Another is that, should paraconsistency be ruled out, then everything would be concluded from a contradiction (meaning that everything is significant, which is antinomic).

The issue of paraconsistency naturally leads us to that of analyzing the standard deduction of the ex falso:

$$\dfrac{\dfrac{\varphi}{\varphi \vee \psi}\,{}^{1} \qquad \neg\varphi}{\psi}\,{}^{2}$$

Relevance logics preclude such a deduction because they rule out disjunctive syllogism (step (2)). A different perspective is to rule out disjunctive weakening (step (1)), as we advocate. The last option is to allow for both inference schemes but to preclude transitivity of inference. This is the path taken by Tennant in his most interesting work on entailment [Tennant,1987]. Unfortunately, failure of transitivity has major drawbacks.

Also, it is worth paying attention to the interaction between disjunction and implication embodied by the formula $(\varphi \vee \psi) \rightarrow \chi$. Let us repeat that, for us, $p \vee q$ really is an alternative between p and q so that $p \vee q$ is actually about p as well as about q. For this

reason, $(\varphi \vee \psi) \rightarrow \chi$ is specifically meant to deduce χ from $\varphi \vee \psi$ whereas $\varphi \rightarrow \chi$ and $\psi \rightarrow \chi$ serve the same purpose with respect to φ and ψ. Accordingly, $(\varphi \vee \psi) \rightarrow \chi$ together with φ, or similarly together with ψ, does not make χ to be deduced. When thinking of it, all this is the more sensible and only is the prejudice attached to unlimited disjunctive weakening making the usual equivalence of $(\varphi \vee \psi) \rightarrow \chi$ with the couple of formulas $\varphi \rightarrow \chi$ and $\psi \rightarrow \chi$ sound all right.

Of course, the notion of significance is intuitive so not everything is clear-cut about significant conclusions. Nonetheless, the above discussion gives us enough constraints to define a first inference system for significant reasoning.

3 NATURAL DEDUCTION

The reader familiar with natural deduction can skip this section.

Throughout the text, we focus on trees in which each node is labeled with a formula. We call them *trees of formulas* (actually, we rather tend to identify a node with its labeling formula).

It is assumed that every link (in fact, hyper-link) relating a parent node to its children is uniquely determined by a so-called *identifier* (which we take to be a natural number).

Given a tree of formulas, we also say that a formula A is an *hypothesis* for a node N iff A is a leaf in the subtree rooted at N. By abuse of language, we say that A is an hypothesis for B when B is the labeling formula of N.

We assume a notational device called *discharge* such that any leaf (in a tree of formulas) can be marked as "discharged". Such a leaf is said to be a discharged formula, as is usual, even though a discharged formula refers to an occurrence: Not all leaves labeled with the same formula need have the same status regarding discharge.

Intuitively, discharged formulas are auxiliary hypotheses that serve to deduce intermediate conclusions whereas the final conclusion does *not* depend on these formulas. In contrast, the final conclusion depends on all non-discharged hypotheses.

As usual, inference rules are used to form trees of formulas corresponding to a deduction. Here, all the inference rules have the following general form:

$$\frac{A_1 \cdots A_n}{C} \quad \left\{ \begin{array}{l} \text{given } A_1 \text{ and } \cdots \text{ and } A_n, \text{ deduce } C \\ \text{possibly subject to some condition(s)} \end{array} \right.$$

where $A_1, \ldots, A_n$ are the *assumptions* of the rule and C is the *consequence* of the rule.

Definition 1 *A derivation Π of a formula C from a set of formulas $\mathcal{P}$ is a finite tree of formulas such that:*

- *The root node of Π is C.*
- *Each leaf of Π is either a discharged formula or a formula in $\mathcal{P}$ (or an axiom, if any).*
- *Each node B of Π has $A_1, \ldots, A_n$ as its child nodes only if there exists an inference rule of which $A_1, \ldots, A_n$ are the assumptions and B the consequence (all constraints, if any, attached to the rule must be met).*
- *Discharge marks in Π only occur as per the stipulations $(\S_n)$ stated for $(I \rightarrow)$ and $(E \vee)$ (see below).*

We write $\mathcal{P} \vdash C$ whenever we have such a derivation Π.

In a derivation, every link is thus to be identified with an instance of an inference rule (which in turn is uniquely determined by the identifier of the link under consideration).

At some point, we will need to consider *normal derivations* [Prawitz,1965] where the intuitive meaning of a normal derivation is that no node in it is both an assumption of the elimination rule yielding its parent node and the conclusion of the corresponding introduction rule yielding one of its child nodes:

Formally, a *maximum segment* in a derivation is a sub-branch $\{A_1, \ldots, A_k\}$ (starting with A_1 and ending with A_k) such that

1. A_1 is the consequence of an introduction rule.
2. For each $i < k$, A_i is a minor assumption of a $(E \vee)$ rule.
3. A_k is a major assumption of an elimination rule.

A_k is a *maximum formula* and k is the *length* of the maximum segment.

A closer look at the following inference system reveals that the introduction and elimination rule referred to

in 1. and 3. are necessarily applied to the same connective.

Finally, a derivation is *normal* iff it contains no maximum segment.

4 NATURAL DEDUCTION FOR SIGNIFICANT REASONING

We adopt a restricted set of logical symbols: $\bot$ (absurdity), $\vee$ (disjunction), $\rightarrow$ (implication). First of all, there is no negation. Formulas $\varphi \rightarrow \bot$ are then used to overcome the absence of explicit negation. Second, there is no conjunction. Instead, sets of formulas are to be interpreted conjunctively. Then, the disjunction of two sets of formulas $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ and $\Psi = \{\psi_1, \ldots, \psi_m\}$ can be given the form of the set

$$\{\varphi_i \vee \psi_j \mid 1 \leq i \leq n, 1 \leq j \leq m\} .$$

And, $\varphi_1 \rightarrow (\ldots \rightarrow (\varphi_n \rightarrow \psi_i)\ldots)$ (for $i = 1..m$) is the way for a series of formula to state that each ψ_i (for $i = 1..m$) follows from $\varphi_1, \ldots, \varphi_n$ taken together (in any order, actually).

In order to avoid heavy notation, we follow the convention that $\vee$ binds stronger than $\rightarrow$.

We adapt a system due to Prawitz [1965] with its notion of a *minor assumption* for an inference rule (it is either of φ, ψ_1, ψ_2 in the list below) and of a *major assumption* for an inference rule (any other assumption).

Our system consists of the following five inference rules:

Introduction rules

$$(I \rightarrow) \frac{\psi_0}{\varphi_0 \rightarrow \psi_0} \ (\S_0)$$

$$(I_l \vee) \frac{\varphi_1}{\varphi_1 \vee \varphi_2} \ (\dagger) \qquad (I_r \vee) \frac{\varphi_2}{\varphi_1 \vee \varphi_2} \ (\dagger)$$

Elimination rules

$$(E \rightarrow) \frac{\varphi \quad \varphi \rightarrow \psi}{\psi}$$

$$(E \vee) \frac{\varphi_1 \vee \varphi_2 \quad \psi_1 \quad \psi_2}{\psi} \ (\ddagger)\,(\S_1)$$

Simplification rule

$$(E_l \ \vee\bot) \frac{\psi \vee \bot}{\psi} \qquad (E_r \ \vee\bot) \frac{\bot \vee \psi}{\psi}$$

The symbols $(\dagger)$, $(\ddagger)$, and $(\S_n)$ indicate that the rules are subject to the following provisos:

Proviso $(\dagger)$: It must be the case that $\varphi_1 \vee \varphi_2$ is a minor assumption of $(E \vee)$.

Proviso $(\ddagger)$: It must be the case that $\psi = \psi_1 = \psi_2$. Moreover, if ψ_1 and ψ_2 are conclusions of $(I\vee)$ rules then one of these must be a $(I_l\vee)$ rule and the other must be a $(I_r\vee)$ rule.

Proviso $(\S_n)$: For $i = n..2n$, it must be the case that φ_i is an hypothesis for ψ_i. Then, any occurrence of φ_i as an hypothesis for ψ_i inherits the identifier of the current instance of the inference rule as its discharge mark.

The application of these provisos is discussed in Section 5.

Axioms for associativity of $\vee$ can be introduced freely:

$$((\varphi_0 \vee \varphi_1) \vee \varphi_2) \rightarrow (\varphi_0 \vee (\varphi_1 \vee \varphi_2))$$

and

$$(\varphi_0 \vee (\varphi_1 \vee \varphi_2)) \rightarrow ((\varphi_0 \vee \varphi_1) \vee \varphi_2) .$$

A classical extension for the above intuitionistic version can be obtained by adding the axiom for excluded middle: $\varphi \vee (\varphi \rightarrow \bot)$.

Observe that all introduction and elimination rules are standard except for the provisos. However, the above system is unusual in several respects. First, there is a $\bot$-related simplification rule for $\vee$. Second, as can be seen from the examples in Section 5, subtrees in a derivation, when taken in isolation, need not be derivations themselves.

5 EXAMPLES AND MAIN RESULT

An illustrative derivation involving discharge is the following one:

$$\dfrac{\dfrac{\dfrac{{}^{(1)}\varphi \quad \varphi \rightarrow \psi}{\psi}\,(E\rightarrow) \quad \dfrac{{}^{(1)}\varphi \quad \varphi \rightarrow (\psi \rightarrow \bot)}{\psi \rightarrow \bot}\,(E\rightarrow)}{\bot}\,(E\rightarrow)}{\varphi \rightarrow \bot}\,(1)\,(I\rightarrow)$$

The derivation "starts" with letting $\varphi \to \psi$ and $\varphi \to (\psi \to \bot)$ as well as two occurrences of φ (at first, ignore the superscript in front of these) be hypotheses (for the time being). Now, the rule $(E \to)$ "is applied" to yield ψ on the one hand and $\psi \to \bot$ on the other. Then, $(E \to)$ is applied once more to yield $\bot$. Next, the rule $(I \to)$ is applied to yield $\varphi \to \bot$. All in all, the final conclusion $\varphi \to (\varphi \to \bot)$ depends on the non-discharged hypotheses, which are $\varphi \to \psi$ and $\varphi \to (\psi \to \bot)$. As for notation, we write $\varphi \to \psi, \varphi \to (\psi \to \bot) \vdash \varphi \to (\varphi \to \bot)$.

The fact that the occurrences of φ are discharged when applying the rule $(I \to)$ is indicated by (1) next to the $(I \to)$ bar and by (1) as a superscript in front of the occurrences of φ. That is, (1) is the identifier for that instance of $(I \to)$ and is also the discharge mark for the occurrences of φ.

Still regarding discharge, the proviso for the $(I \to)$ rule discriminates

$$\frac{{}^{(1)}\varphi}{\varphi \to \varphi}\ {\scriptstyle (1)\ (I\to)}$$

which is a derivation in our system, from

$$\frac{\psi}{\varphi \to \psi}\ {\scriptstyle (I\to)}$$

which is not derivable in our system.

The proviso for the $(I\vee)$ rule lets

$$\frac{\psi \vee \varphi \qquad \dfrac{{}^{(1)}\psi}{\varphi \vee \psi}\ {\scriptstyle (I_r\vee)} \qquad \dfrac{{}^{(1)}\varphi}{\varphi \vee \psi}\ {\scriptstyle (I_l\vee)}}{\varphi \vee \psi}\ {\scriptstyle (1)\ (E\vee)}$$

be a derivation in our system, whereas

$$\frac{\varphi}{\varphi \vee \psi}\ {\scriptstyle (I_l\vee)}$$

is not derivable in our system.

For the record, here is a derivation of disjunctive syllogism:

$$\frac{\varphi \vee \psi \qquad \dfrac{\dfrac{{}^{(1)}\varphi \quad \varphi \to \bot}{\bot}\ {\scriptstyle (E\to)}}{\psi \vee \bot}\ {\scriptstyle (I_r\vee)} \qquad \dfrac{{}^{(1)}\psi}{\psi \vee \bot}\ {\scriptstyle (I_l\vee)}}{\psi \vee \bot}\ {\scriptstyle (1)\ (E\vee)}$$
$$\frac{\psi \vee \bot}{\psi}\ {\scriptstyle (E_l\vee\bot)}$$

Apart from disjunctive syllogism, the following derivable inferences are of interest:

$$\frac{\varphi_1 \vee \varphi_2 \qquad \varphi_1 \to \psi_1 \qquad \varphi_2 \to \psi_2}{\psi_1 \vee \psi_2}$$

$$\frac{\varphi \to (\psi \to \chi)}{\psi \to (\varphi \to \chi)}$$

$$\frac{\varphi \to \psi \qquad \psi \to \chi}{\varphi \to \chi}$$

$$\frac{\varphi \to \psi \qquad \psi \to \bot}{\varphi \to \bot}$$

In fact, the last inference is modus tollens. Similarly, one may derive all other forms of reasoning by contraposition.

As common with natural deduction systems, the most fundamental property is that of normalization (because it corresponds to cut-elimination in sequent calculi):

Theorem 1 (Normalization) *Every derivation can be transformed into a normal derivation (with the same premises and conclusion).*

In our case, normalization yields additionally the following salient features:

Corollary 1 (Transitivity) $\varphi \vdash \psi$ *and* $\psi \vdash \chi$ *implies* $\varphi \vdash \chi$ *for all* φ, ψ, χ.

Corollary 2 (Paraconsistency) *For all* φ *there exists* ψ *such that* $\varphi, \varphi \to \bot \nvdash \psi$.

Corollary 2 is usually expressed as $\varphi, \neg\varphi \nvdash \psi$.

Furthermore, observe that

$$\begin{aligned} \varphi \vee \psi, \varphi \to \bot &\vdash \psi, \\ \varphi, \varphi \to \bot &\nvdash \psi, \\ \varphi, \varphi \vee \psi, \varphi \to \bot &\vdash \psi. \end{aligned}$$

6 DISCUSSION

All rules in our system are either restricted forms of rules for classical logic or rules derivable in classical logic. Thus, our system is a subsystem of natural

deduction for classical logic. Let us look at some of the inferences that are no longer derivable.

First, tautologies corresponding to conditionalization and disjunctive weakening are no longer derivable in our system:

$$\not\vdash \varphi \to (\varphi \vee \psi),$$

$$\not\vdash \varphi \to (\psi \to \varphi).$$

The latter comes together with the non-derivability of the classical tautology

$$\not\vdash (\varphi \to \psi) \vee (\psi \to \varphi),$$

for which it has then no reason to be supported, since it relies on arbitrary conditionalization.

As was to be expected from the discussion at the end of Section 2,

$$\frac{\varphi_1 \quad (\varphi_1 \vee \varphi_2) \to \psi}{\psi}$$

is not derivable. Analogously, the following inference is not valid in our system:

$$\frac{\varphi_1 \quad (\varphi_2 \to \varphi_1) \to \psi}{\psi}$$

Similarly to the case of disjunction, where $(\varphi \vee \psi) \to \chi$ is not related to $\varphi \to \chi$ and $\psi \to \chi$, we must therefore distinguish between $(\psi \to \varphi) \to \chi$ and $\varphi \to \chi$. In accord with the discussion at the end of Section 2, we differentiate between a formula q and a formula $p \to q$ that links information about q with that about p. Therefore, we also distinguish between inferences drawn from q and $p \to q$.

The most surprising case is presumably the failure of

$$\frac{\varphi \to \psi}{\varphi \to (\varphi \to \psi)}$$

but more relaxed technical conditions may turn it into a valid scheme (this is further discussed in the conclusion).

Adapting traditional natural deduction had us make an implicit decision and that is that the system is monotonic although this property did not arise from our discussion about significant reasoning. Consider a tautology such as $p \to p$. It certainly is significant information to conclude when no premise is given. This is no longer the case, should the premise p be given. Does this mean that $p \to p$ must then be withdrawn? We think it would be too strict a principle. Rather, we prefer to consider as significant any conclusion that could be viewed as such for *some* reason (in particular, in light of *part* of the given premises). The philosophy here would rather be that we can dispense with drawing some inferences but not to rule them out altogether.

Our system seems promising whenever it comes to knowledge representation problems involving (some extent of) relevance. Let us illustrate this by considering a classically valid yet counterintuitive inference that has been identified by Stephen Read in [Read,1989].

> "Roy has claimed that John was in Edinburgh on a certain day, and Crispin has denied it."

Now, consider

1. "If John was in Edinburgh, Roy was right."
2. "It is not the case that if Crispin was right, so was Roy."
3. "If John was in Edinburgh, Crispin was right."

The latter sentence is false but the former two are true. This gives us an invalid argument (all its premises are true and its conclusion is false). However, the argument is classically valid because

$$\varphi \to \psi, \neg(\chi \to \psi) \vdash \varphi \to \chi$$

holds in classical logic. To see this, consider the following derivation.

$$\dfrac{\dfrac{\dfrac{(\chi \to \psi) \to \bot \qquad \dfrac{\dfrac{^{(3)}\varphi \quad \varphi \to \psi}{\psi}\,(E\to)}{\chi \to \psi}\,(1)\,(I\to)}{\bot}\,(E\to)}{(\chi \to \bot) \to \bot}\,(2)\,(I\to)}{\chi}\,(E\neg\neg)}{\varphi \to \chi}\,(3)\,(I\to)$$

This derivation is invalid in our systems since it violates proviso ($\S_0$) at (1) ($I \to$) and (2) ($I \to$). In a classical natural deduction system the inference

$\varphi \to \psi, \neg(\chi \to \psi) \vdash \varphi \to \chi$ is always established by applying arbitrary conditionalization, which is disallowed in our system as in relevance logics.

However, relevance logics only overlap with the requirements of a notion of significant inference. On the one hand, disjunctive syllogism is valid in our system but is invalid in relevance logics. On the other hand, disjunctive weakening is invalid in our system but is valid in relevance logics (including first-degree entailment [Anderson and Belnap,1975]).

Our system also bears some connections with relatedness [Epstein,1979; Krajewski,1986] but it is closer to Parry's [1989] and even closer to Tennant's [1987]. However, Tennant is strongly concerned with relevance when implication is involved and he ignores the matter when it comes to disjunction. Also, Tennant's system fails transitivity but ours satisfies it (namely, $\varphi \vdash \psi$ and $\psi \vdash \chi$ yields $\varphi \vdash \chi$).

As already mentioned, negation can be introduced in our system by way of the usual implication to absurdity. In fact, if we take

$$\neg\varphi \stackrel{\text{def}}{=} \varphi \to \bot$$

then the usual inferences hold (except that the degenerate case where $\bot$ does not depend on ${}^{(1)}\varphi$ is not allowed here):

$$\dfrac{\begin{array}{c}{}^{(1)}\varphi \\ \vdots \\ \bot\end{array}}{\varphi \to \bot}\,(1)\,(I\to) \quad \Longleftrightarrow \quad \dfrac{\begin{array}{c}{}^{(1)}\varphi \\ \vdots \\ \bot\end{array}}{\neg\varphi}\,(1)\,(I\neg)$$

and

$$\dfrac{\varphi \to \bot \quad \varphi}{\bot}\,(E\to) \quad \Longleftrightarrow \quad \dfrac{\neg\varphi \quad \varphi}{\bot}\,(E\neg)$$

Turning to conjunction,

$$\varphi \wedge \psi \stackrel{\text{def}}{=} ((\varphi \to \bot) \vee (\psi \to \bot)) \to \bot$$

only the usual intuitionistic inference holds, as shown in Figure 1, and the result is not as meaningful as in the case of negation, since the corresponding elimination rule is not derivable in our system. As with intuitionistic logic [Dummett,1977], there is no way of deriving φ from $\neg(\neg\varphi \vee \neg\psi)$. Among others, this is a reason why we advocate modeling conjunctions by appeal to sets, as described at the start of Section 4.

7 CONCLUSION

With significant reasoning, we have elaborated upon a new notion of reasoning that is distinct from existing approaches, although there are close ties to relevance logic and intuitionistic logic.

Our contribution can thus be looked at from two perspectives: First, we have identified and elaborated upon the notion of significant reasoning. And second, we have defined a version of natural deduction for significant reasoning.

Of course, the notion of significance is intuitive so that not everything is clear-cut about significant conclusions. There are several places where another choice could make sense. In fact, our choices were motivated by the strict realization of our intuitions discussed in Section 2. For instance, we could be less strict about the policy for discharging hypotheses so that $\varphi \to \psi$ yields $\varphi \to (\varphi \to \psi)$. (Although this is characteristic for linear logic [Girard,1987], it should be clear that our system has nothing in common with resource logics.) Also, we could consider to get closer to familiar practice so that φ and $(\varphi \vee \psi) \to \chi$ yield χ (although we still disapprove of such an inference).

As discussed in Section 5, our system seems promising whenever it comes to knowledge representation problems involving (some extent of) relevance. As pointed out by one of the anonymous referees, this is of interest for the definition of reasoning capabilities for agents in a multi agent systems.

An important issue of future research consists of elaborating appropriate semantical underpinnings. A promising starting point could be to adapt the semantics of relatedness logic [Epstein,1979]. That is, models would be equipped with relations reflecting a notion of significance among propositions.

A SOME TECHNICALITIES

Let $\mathcal{F}$ be the set of all formulas of the language.

A *tree of formulas* is a triple $\langle I, T, f\rangle$ where

1. I is a finite initial portion of the numerals $\{\overline{1}, \overline{2}, \overline{3}, \ldots\}$ for the natural numbers
2. T is a subset of the free monoid I^* such that:
   (a) if $uv \in T$ then $u \in T$
   (b) if $u\overline{n} \in T$ then $u\overline{m} \in T$ for $m < n$
3. f is a function from T to $\mathcal{F}$

$$\dfrac{\dfrac{^{(2)}((\varphi \to \bot) \vee (\psi \to \bot)) \qquad \dfrac{^{(1)}(\varphi \to \bot) \quad \varphi}{\bot} \qquad \dfrac{^{(1)}(\psi \to \bot) \quad \psi}{\bot}}{\bot}\,{}^{(1)}\,(E\vee)}{((\varphi \to \bot) \vee (\psi \to \bot)) \to \bot}\,{}^{(2)}\,(I\to) \quad \Longleftrightarrow \quad \dfrac{\varphi \quad \psi}{\varphi \wedge \psi}\,(I\wedge)$$

Figure 1: Derivation of adjunction.

Each $w \in T$ is a *node* of the tree. The tree is *finite* iff it only has finitely many nodes (i.e., T is finite). The empty tree has no node (i.e., $T = \emptyset$). The empty word ε is the *root* of the tree (on condition that the tree is non-empty, $T \neq \emptyset$). A *leaf* of the tree is a node w such that $w\overline{1} \notin T$. If $w \in T$ and $w\overline{n} \in T$ then $w\overline{n}$ is a *child* node of w and w is the *parent* node of $w\overline{n}$.

A *branch* of the tree $\langle I, T, f\rangle$ is any $B \subseteq T$ such that:

1. $\varepsilon \in B$
2. if $u \in B$ and $u\overline{1} \in T$ then there exists exactly one $\overline{n} \in I$ such that $u\overline{n} \in B$

Given a branch B of the tree $\langle I, T, f\rangle$, a *sub-branch* starting with $u \in B$ and ending with $v \in B$ is any non-empty $B' \subseteq B$ such that $w \in B$ is a member of B' iff w is a (possibly improper) suffix of u and v is a (possibly improper) prefix of w.

Let $\langle I, T, f\rangle$ with $u \in T$. The *subtree rooted at* u is the tree $\langle I, T', f'\rangle$ where $T' = \{v \mid uv \in T\}$ and $f'(v) = f(uv)$ for all $v \in T'$.

B PROOF OF NORMALIZATION, PARACONSISTENCY, AND TRANSITIVITY

Theorem 1 (Normalization) *Every derivation can be transformed into a normal derivation (with the same premises and conclusion).*

Corollary 1 (Transitivity) $\varphi \vdash \psi$ *and* $\psi \vdash \chi$ *implies* $\varphi \vdash \chi$ *for all* φ, ψ, χ.

Corollary 2 (Paraconsistency) *For all* φ *there exists* ψ *such that* $\varphi, \neg\varphi \nvdash \psi$.

We prove that if a formula ψ concludes a derivation in which a maximal formula φ occurs then there exists a normal derivation of ψ.

We first show how to reduce the length of maximum segments when necessary: Considering a maximum segment whose maximum formula is lowest in a branch (cf. the lowest occurrence of T below), apply the following permutation

$$\dfrac{\dfrac{\overset{\Delta}{X \vee Y} \quad \overset{\Sigma_1}{T} \quad \overset{\Sigma_2}{T}}{T}}{\begin{matrix}\Pi\\ Z\\ \Theta\end{matrix}} \quad \Longrightarrow \quad \dfrac{\overset{\Sigma_1}{T} \quad \overset{\Sigma_2}{T}}{\dfrac{\overset{\Delta}{X \vee Y} \quad \overset{\Pi}{Z} \quad \overset{\Pi}{Z}}{Z}}\\ \Theta$$

In the resulting derivation, there can be no maximal segment through $X \vee Y$. Therefore, there can be no new maximum segment in the leftmost branch. Similarly with the other two branches. However, the length of the maximum segment we considered is now decreased by one. The process can be iterated until the desired length is obtained for any maximal segment(s) we consider.

When considering a maximal segment of length 1 whose maximal formula is $X \to Y$, we apply the following transformation:

$$\dfrac{\dfrac{\begin{matrix}^{(1)}X\\ \Sigma_2\\ Y\end{matrix}}{X \to Y}\,{}^{(1)}\,(I\to) \quad \overset{\Sigma_1}{X}}{\begin{matrix}Y\\ \Pi\\ Z\end{matrix}} \quad \Longrightarrow \quad \begin{matrix}\Sigma_1\\ X\\ \Sigma_2\\ Y\\ \Pi\\ Z\end{matrix}$$

Note that all formulas discharged in Σ_1 or Σ_2 are still discharged in the resulting derivation (if n occurrences of X are discharged within Σ_2 then the resulting derivation is to display n additional copies of Σ_1). Of course, no new discharge is introduced. Clearly, every other proviso of the rules is also satisfied. Therefore, we have obtained a derivation with the same hypotheses and the same conclusion.

Observe that there can be a new maximal segment (in the resulting derivation) only if X or Y is involved in it. Further observe that there can be a new maximal segment involving X only if either X or a formula in Σ_2 is a maximal formula. Whatever is the case, the definition of a maximal segment forces the new

maximal formula to be an occurrence of X. The same applies to Y wrt Π. All in all, the (at most two) new maximal formulas X and Y are sub-formulas of the initial maximal formula $X \to Y$.

When considering a maximal segment of length 2 (cf. (†)) whose maximal formula is $X \vee Y$, we apply the following transformation:

$$\dfrac{\dfrac{\Delta \quad \overset{\Sigma_1}{X} \quad \overset{\Sigma_2}{Y} \quad \overset{(1)}{X} \quad \overset{(1)}{Y}}{W \quad X \vee Y \quad X \vee Y \quad \Theta_1 \quad \Theta_2}}{\dfrac{X \vee Y \qquad T \qquad T}{T}\ (1)\ (E\vee)}$$
$$\Pi$$
$$Z$$

$$\Longrightarrow \quad \dfrac{\dfrac{\overset{\Sigma_1}{X} \quad \overset{\Sigma_2}{Y}}{\Delta \quad \Theta_1 \quad \Theta_2}}{W \quad T \quad T}$$
$$T$$
$$\Pi$$
$$Z$$

Note that all formulas discharged in Σ_1 or Σ_2 are still discharged in the resulting derivation (observe that, in the original derivation, discharging within any Θ_i an hypothesis introduced in some Σ_j is incorrect). Also, all formulas introduced in Θ_1 and Θ_2, if discharged, are still discharged in the resulting derivation. Similarly, every other proviso of the rules is also satisfied. Again, we obtain a derivation with the same hypotheses and the same conclusion.

Observe that there can be a new maximal segment (in the resulting derivation) only if X or Y is involved in it. Further observe that there can be a new maximal segment involving X only if either X or a formula in Θ_1 is a maximal formula. Whatever is the case, the definition of a maximal segment forces the new maximal formula to be an occurrence of X. The same applies to Y wrt Θ_2. All in all, the (at most two) new maximal formulas X and Y are sub-formulas of the initial maximal formula $X \vee Y$.

In the resulting derivation, only X and Y can be new maximal formulas (whether the initial maximal formula is $X \to Y$ or $X \vee Y$). So, the transformation either decreases the number of maximal formulas or replaces a maximal formula φ by simpler maximal formulas (actually, one or two sub-formulas of φ). Of course, formulas have a finite number of occurrences of the connectives and atomic formulas never are maximal formulas. Repeatedly applying the transformation is then a finite process, ending with a derivation in which no maximal formula occurs.

For the purpose of applying this theorem together with a result due to Tennant, we consider the case where the $(E\vee)$ rule followed by the simplification rule can be normalized in a special $(E\vee)$ rule as follows: When some minor assumption ψ_1 or ψ_2 of $(E\vee)$ is $\bot$, then the conclusion ψ is a copy of the other minor assumption. Clearly, the above transformation still gives us the desired outcome and the proof is over.

In view of the system defined by Tennant in [1987, p. 672], this theorem yields the desired result that our system is paraconsistent.

Acknowledgements

This work was supported by the French foreign ministry within programme PROCOPE under grant number 99027 and the German ministry for education, science, research and technology within programme PROCOPE under grant number 9822866.

References

[Anderson and Belnap, 1975] A. Anderson and N. Belnap. *Entailment: The Logic of Relevance and Necessity, Vol. I.* Princeton University Press, 1975.

[Dummett, 1977] M. Dummett. *Elements of intuitionism.* Oxford University Press, 1977.

[Epstein, 1979] R. Epstein. Relatedness and implication. *Philosophical Studies*, 36:137–173, 1979.

[Gentzen, 1935] G. Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210 and 405–431, 1935. English translation in [Szabo,1969].

[Girard, 1987] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Krajewski, 1986] S. Krajewski. Relatedness logic. *Report on Mathematical Logic*, 20:7–14, 1986.

[Parry, 1989] W. Parry. Analytic implication: Its history, justification, and varieties. In R. Routley and J. Norman, editors, *Directions in Relevant Logic.* Nijhoff, Amsterdam, 1989.

[Prawitz, 1965] D. Prawitz. *Natural Deduction.* Alınqvist & Wiksell, Stockholm, 1965.

[Read, 1989] S. Read. *Relevant Logic. A Philosophical Examination of Inference.* Basil Blackwell, 1989.

[Robinson, 1965] J. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, 1965.

[Szabo, 1969] M. Szabo, editor. *The collected papers of Gerhard Gentzen*. North-Holland, 1969.

[Tennant, 1987] N. Tennant. Natural deduction and sequent calculus for intuitionistic relevant logic. *Journal of Symbolic Logic*, 52(3):665–680, 1987.

Unfolding Partiality and Disjunctions in Stable Model Semantics

Tomi Janhunen, Ilkka Niemelä, Patrik Simons
Lab. for Theoretical Computer Science
Dept. of Computer Science and Eng.
Helsinki University of Technology
P.O.Box 5400, 02015 HUT, Finland
{Tomi.Janhunen,Ilkka.Niemela,Patrik.Simons}@hut.fi

Jia-Huai You
Dept. of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2H1
you@cs.ualberta.ca

Abstract

The paper studies an implementation methodology for partial and disjunctive stable models where partiality and disjunctions are unfolded from a logic program so that an implementation of stable models for normal (disjunction-free) programs can be used as the core inference engine. The unfolding is done in two separate steps. Firstly, it is shown that partial stable models can be captured by total stable models using a simple linear and modular program transformation. Hence, reasoning tasks concerning partial models can be solved using an implementation of total models. Disjunctive partial stable models have been lacking implementations which now become available as the translation handles also the disjunctive case. Secondly, it is shown how total stable models of disjunctive programs can be determined by computing stable models for normal programs. Hence, an implementation of stable models of normal programs can be used as a core engine for implementing disjunctive programs. The feasibility of the approach is demonstrated by constructing a system for computing stable models of disjunctive programs using the *smodels* system as the core engine. The performance of the resulting system is compared to that of *dlv* which is a state-of-the-art special purpose system for disjunctive programs.

1 INTRODUCTION

Implementation techniques for declarative semantics of logic programs have advanced considerably during the last years. For example, the XSB system (Sagonas, Swift and Warren 1996) is a WAM-based full logic programming system supporting the well-founded semantics. In addition to this kind of a skeptical approach that is based on query evaluation also a credulous approach focusing on computing models of logic programs is gaining popularity. This work has been centered around the stable model semantics (Gelfond and Lifschitz 1988). There are reasonably efficient implementations available for computing stable models for disjunctive and normal (disjunction-free) programs, e.g., *dlv* (Leone et al. 1999), *DeReS* (Cholewiński, Marek and Truszczyński 1996), and *smodels* (Niemelä and Simons 1997, Simons 1999b). Furthermore, there are interesting applications in areas such as planning (Dimopoulos, Nebel and Koehler 1997), model checking (Liu, Ramakrishnan and Smolka 1998), reachability analysis (Heljanko 1999), and constraint satisfaction (Niemelä 1999). This approach is emerging as a new paradigm for logic programming (Marek and Truszczyński 1999, Niemelä 1999).

This paper aims to widen the scope of the stable model paradigm. It addresses two issues: partial models and disjunctions. Sometimes it is natural to use partial models to represent a domain. Even when working with total models, partial models could be useful, e.g., for debugging purposes to show what is wrong in a program without any total models. However, little has been done on implementing partial model computation and most of the work has focused on query evaluation w.r.t. the well-founded semantics. In the paper we show that total stable models can capture partial stable models using a simple linear program transformation. This transformation works also in the disjunctive case showing that implementations of total stable models, e.g. *dlv*, can be used for computing partial stable models. Using a suitable transformation of queries, a mechanism for query answering can be realized as well.

Our translation is interesting in many respects. First, it should be noted that the translation does not follow directly from the complexity results already available. It has been shown, e.g., that the problem of deciding whether a query is contained in some model (possibility inference) is Σ_2^P-complete for both partial stable models and total stable models (Eiter and Gottlob 1995, Eiter, Leone and Saccà 1998). This implies that there exists a polynomial time reduction from possibility inference w.r.t. partial models to possibility inference w.r.t. total models. However, this kind of a translation is guaranteed to preserve only the yes/no answer to the possibility inference problem. To the best of our knowledge, there is no program transformation given in the literature which captures partial stable models in terms of stable models. Second, not all translations are sufficient from a computational point of view. In practice, when a program is compiled into another form to be executed, certain computational properties of the translation play an important role:

- efficiency of the compilation (in which order of polynomial),
- modularity (are independent, separate compilations of parts of a program possible), and
- structural preservation (are the composition and intuition of the original program preserved so that debugging and understanding of runtime behavior are made possible).

All this points to the importance of finding good translation methods to enable the use of an existing inference engine to solve other interesting problems.

The efficiency of procedures for computing stable models of normal programs has increased substantially in recent years. This raises the question whether such an implementation can be used as a *core engine* for implementing other reasoning systems. We study the feasibility of this approach by developing a method for computing stable models of disjunctive programs using such a core engine. This is non-trivial as deciding whether a disjunctive program has a stable model is Σ_2^p-complete whereas the problem is NP-complete in the non-disjunctive case. The method has been implemented using *smodels* as the core engine. The performance of the implementation is compared to that of *dlv* that is a state-of-the-art special purpose system for disjunctive programs.

There are a number of novelties in the work. Maximal partial stable models for normal programs are known as *regular models*, *M-stable models*, and *preferred extensions* (Dung 1995, Saccà and Zaniolo 1990, You and Yuan 1994). Although this semantics has a sound and complete top-down query answering procedure (Dung 1995, Eshghi and Kowalski 1989), so far very little effort has been given to a serious implementation. For disjunctive programs, to our knowledge, no implementation has ever been attempted. As a result, we obtain (perhaps) the first scalable implementation of the regular model/preferred extension semantics, and the first implementation ever for partial stable model semantics for disjunctive programs. Our technical work on the relationship between stable and partial stable models via a translational approach provides a compelling argument for the naturalness of partial stable models: stable models and partial stable models share the same notion of unfoundedness, carefully studied earlier in (Eiter, Leone and Saccà 1997, Leone, Rullo and Scarcello 1997). In fact, our work extends the previous understanding in the following way: stable and partial stable models capture the same semantics by different representing programs.

The rest of the paper is structured as follows. We first review the basic definitions and concepts and then show that partial models can be captured using a simple program transformation. In Section 4 we describe the method for computing disjunctive stable models using an implementation of non-disjunctive programs as a core engine. Then we present some experimental results and finish with some concluding remarks.

2 DEFINITIONS AND NOTATIONS

A *disjunctive logic program* P (or, just *disjunctive program* P) is a set of rules of the form

$$a_1 \vee \cdots \vee a_k \leftarrow b_1, \ldots, b_m, \sim c_1, \ldots, \sim c_n \qquad (1)$$

where $k \geq 1$, $m, n \geq 0$ and a_i's, b_i's and c_i's are atoms from the Herbrand base $\mathrm{Hb}(P)$[1] of P. Let us also distinguish subclasses of disjunctive programs. If $k = 1$ for each rule of P, then P is a *disjunction-free* or *normal program*. If $n = 0$ for each rule of P, then P is called *positive*.

Literals are either atoms from $\mathrm{Hb}(P)$ or expressions of the form $\sim a$ where $a \in \mathrm{Hb}(P)$. For a set of atoms $A \subseteq \mathrm{Hb}(P)$, we define $\sim A$ as $\{\sim a \mid a \in A\}$. Let us introduce a shorthand $A \leftarrow B, \sim C$ for rules where $A \neq \emptyset$, B and C are subsets of $\mathrm{Hb}(P)$. In harmony with (1), the set of atoms A in the *head* of the rule is interpreted disjunctively while the set of literals $B \cup \sim C$ in the *body* of the rule is interpreted conjunctively. We wish

[1] For the sake of convenience, we assume that a given program P is already instantiated by the underlying Herbrand universe, and is thus ground.

to further simplify the notation $A \leftarrow B, \sim C$ in some particular cases. When A, B or C is a singleton $\{a\}$, we write a instead of $\{a\}$. If $B = \emptyset$ or $C = \emptyset$ we omit B and $\sim C$ (respectively) as well as the separating comma in the body of the rule.

2.1 PARTIAL AND TOTAL MODELS

Let P be any disjunctive program. A *partial interpretation* I for P is a pair $\langle T, F \rangle$ of subsets of $\mathrm{Hb}(P)$ such that $T \cap F = \emptyset$. The atoms in the sets $I^{\mathbf{t}} = T$, $I^{\mathbf{f}} = F$ and $I^{\mathbf{u}} = \mathrm{Hb}(P) - (T \cup F)$ are considered to be *true*, *false*, and *undefined*, respectively. We introduce constants $\mathbf{t}$, $\mathbf{f}$, and $\mathbf{u}$, to denote the respective three truth values. A partial interpretation I for P is a *total* interpretation for P whenever $I^{\mathbf{u}} = \emptyset$, i.e., if every atom of $\mathrm{Hb}(P)$ is either true or false. When no confusion arises, we use $I^{\mathbf{t}}$ alone to specify a total interpretation I for P (then $I^{\mathbf{f}} = \mathrm{Hb}(P) - I^{\mathbf{t}}$ and $I^{\mathbf{u}} = \emptyset$ hold).

Given a partial interpretation for P, the truth values of atoms are determined by $I^{\mathbf{t}}$, $I^{\mathbf{f}}$ and $I^{\mathbf{u}}$ as explained above while $\mathbf{t}$, $\mathbf{f}$ and $\mathbf{u}$ have their fixed truth values. For more complex logical expressions E, we use $I(E)$ to denote the truth value of E in I. The value $I(\sim a)$ is defined to be $\mathbf{t}$, $\mathbf{f}$, or $\mathbf{u}$ whenever $I(a)$ is $\mathbf{f}$, $\mathbf{t}$, or $\mathbf{u}$, respectively. To handle conjunctions and disjunctions, we introduce an ordering on the three truth values by setting $\mathbf{f} < \mathbf{u} < \mathbf{t}$. By default, a set of literals $L = \{l_1, \ldots, l_n\}$ denotes the conjunction $l_1 \wedge \cdots \wedge l_n$ while $\bigvee L$ denotes the corresponding disjunction $l_1 \vee \cdots \vee l_n$. The truth values $I(L)$ and $I(\bigvee L)$ are defined as the respective minimum and maximum among the truth values $I(l_1), \ldots, I(l_n)$. A rule $A \leftarrow B, \sim C$ is satisfied in I if and only if $I(\bigvee A) \geq I(B \cup \sim C)$. A partial interpretation M for P is a *partial model* of P if all rules of P are satisfied in M, and for a *total model*, also $M^{\mathbf{u}} = \emptyset$ holds. Finally, we introduce an ordering among partial models of a disjunctive program: $M_1 \leq M_2$ iff ${M_1}^{\mathbf{t}} \subseteq {M_2}^{\mathbf{t}}$ and ${M_1}^{\mathbf{f}} \supseteq {M_2}^{\mathbf{f}}$. A partial model M of P is a *minimal* one if there is no partial model M' of P such that $M' < M$ (i.e., $M' \leq M$ and $M' \neq M$).

2.2 STABLE MODELS

Given a partial interpretation I for a disjunctive program P, we define a reduction of P as follows:

$$P^I = \{A \leftarrow B \mid A \leftarrow B, \sim C \in P \text{ and } C \subseteq I^{\mathbf{f}}\}.$$

Note that this transformation coincides with the *Gelfond-Lifschitz reduction* of P (the GL-reduction of P) when I is a total interpretation.

Definition 2.1 (Total stable model) *A total interpretation N for a disjunctive program P is a stable model iff N is a minimal total model of P^N.*

The original definition of partial stable models (Przymusinski 1990) is based on a weaker reduction. Given a disjunctive program P and an interpretation I, the reduction P_I is the set of rules obtained from P by replacing any $\sim c$ in the body of a rule by $I(\sim c)$. As noted in (Przymusinski 1990), the only practical difference between P^I and P_I is that P_I has rules that correspond to rules of $A \leftarrow B, \sim C \in P$ satisfying $I(\sim C) = \mathbf{u}$. Note that if $I(\sim C) = \mathbf{t}$, then $A \leftarrow B \in P^I$, and if $I(\sim C) = \mathbf{f}$, then the partial models of P_I are not constrained by the rule included in P_I.

Definition 2.2 (Partial stable model) *A partial interpretation M for a disjunctive program P is a partial stable model of P iff M is a minimal partial model of P_M.*

(Partial) stable models are intimately related to *unfounded sets* (Eiter et al. 1997, Leone et al. 1997).

Definition 2.3 (Unfounded sets) *Let I be a partial interpretation for a disjunctive program P. A set $U \subseteq \mathrm{Hb}(P)$ of ground atoms is an* unfounded set *for P w.r.t. I, if at least one of the following conditions holds for each rule $A \leftarrow B, \sim C \in P$ such that $A \cap U \neq \emptyset$:*

UF1: $B \cap I^{\mathbf{f}} \neq \emptyset$ *or* $C \cap I^{\mathbf{t}} \neq \emptyset$,

UF2: $B \cap U \neq \emptyset$, *or*

UF3: $(A - U) \cap (I^{\mathbf{t}} \cup I^{\mathbf{u}}) \neq \emptyset$.

An unfounded set U for P w.r.t. I is I-consistent iff $U \cap I^{\mathbf{t}} = \emptyset$.

The first and the last condition coincide with $I(B \cup \sim C) = \mathbf{f}$ and $I(\bigvee (A - U)) \neq \mathbf{f}$, respectively. The intuition is that the atoms of an unfounded set U can be assumed to be false without violating the satisfiability of the rules of the program that contain some atoms from U. In particular, a partial interpretation I for P becomes interesting, if the union of all unfounded sets U for P w.r.t. I is also an unfounded set for P w.r.t. I. In this case, the program P possesses *the greatest unfounded* set U for P w.r.t. I.[2] A total interpretation I is considered to be *unfounded free* for a program P iff there is no unfounded set U for P w.r.t. I such that $U \cap I^{\mathbf{t}} \neq \emptyset$. These notions lead to the following characterization of stable models.

[2] For normal programs P, this property holds for all partial interpretations I; however, for disjunctive programs this is no longer the case.

Theorem 2.4 (Leone et al. 1997) *If M is a total interpretation for a disjunctive program P, then M is a stable model of P iff $M^{\mathbf{f}}$ is the greatest unfounded set for P w.r.t. M iff M is unfounded free for P.*

On the other hand, (Eiter et al. 1997) show that partial stable models can be defined essentially without reference to three-valued logic.

Theorem 2.5 (Eiter et al. 1997) *If M is a partial interpretation for a disjunctive program P, then M is a partial stable model of P iff*

- *$M^{\mathbf{t}}$ is a minimal total model of P^M and*
- *$M^{\mathbf{f}}$ is a maximal M-consistent unfounded set for P w.r.t. M.*

3 UNFOLDING PARTIALITY

Let P be a disjunctive program. In the following, we describe a translation of P into another disjunctive program $\mathrm{Tr}(P)$ such that the stable models of $\mathrm{Tr}(P)$ correspond to the partial stable models of P.

Let us introduce a new atom $a^\bullet$ for each $a \in \mathrm{Hb}(P)$. The intuitive reading of $a^\bullet$ is that a is *potentially* true. For a set of literals $L \subseteq \mathrm{Hb}(P) \cup {\sim}\mathrm{Hb}(P)$, we define $L^\bullet = \{a^\bullet \mid a \in L\} \cup \{{\sim}a^\bullet \mid {\sim}a \in L\}$. The translation $\mathrm{Tr}(P)$ of a disjunctive program P is as follows:

$$\begin{aligned}&\mathrm{Tr}(P) = \\ &\quad \{A \leftarrow B, {\sim}C^\bullet; A^\bullet \leftarrow B^\bullet, {\sim}C \mid A \leftarrow B, {\sim}C \in P\} \cup \\ &\quad \{a^\bullet \leftarrow a \mid a \in \mathrm{Hb}(P)\}\end{aligned}$$

where we use semicolons to separate program rules. Note that the Herbrand base of $\mathrm{Hb}(\mathrm{Tr}(P))$ is $\mathrm{Hb}(P) \cup \mathrm{Hb}(P)^\bullet$. The rules $a^\bullet \leftarrow a$ introduced for each $a \in \mathrm{Hb}(P)$ enforce consistency in the sense that if a is true, then a must also be potentially true. The intended representation of a partial stable model M of P is given by the following equations.

Definition 3.1 *Let M be a partial interpretation of a program P and N a total interpretation of* $\mathrm{Tr}(P)$. *M and N are said to satisfy the* correspondence equations *iff the following equations hold.*

$$M^{\mathbf{t}} = \{a \in \mathrm{Hb}(P) \mid a \in N^{\mathbf{t}} \text{ and } a^\bullet \in N^{\mathbf{t}}\} \quad \text{(CE1)}$$

$$M^{\mathbf{f}} = \{a \in \mathrm{Hb}(P) \mid a \in N^{\mathbf{f}} \text{ and } a^\bullet \in N^{\mathbf{f}}\} \quad \text{(CE2)}$$

$$M^{\mathbf{u}} = \{a \in \mathrm{Hb}(P) \mid a \in N^{\mathbf{f}} \text{ and } a^\bullet \in N^{\mathbf{t}}\} \quad \text{(CE3)}$$

$$\emptyset = \{a \in \mathrm{Hb}(P) \mid a \in N^{\mathbf{t}} \text{ and } a^\bullet \in N^{\mathbf{f}}\} \quad \text{(CE4)}$$

For instance, an atom $a \in \mathrm{Hb}(P)$ is undefined in a partial interpretation M for P exactly whenever a is false and $a^\bullet$ is true in the corresponding total interpretation N for $\mathrm{Tr}(P)$. Note that total interpretations that are models of $\mathrm{Tr}(P)$ satisfy CE4 immediately, since the set of rules $\{a^\bullet \leftarrow a \mid a \in \mathrm{Hb}(P)\}$ is included in $\mathrm{Tr}(P)$. Consequently, the "fourth truth value" is ruled out. The following example demonstrates how the representation given in Definition 3.1 allows us to capture the partial stable models of a disjunctive program P with the total stable models of $\mathrm{Tr}(P)$.

Example 3.2 *Consider a disjunctive program*

$$P = \{a \vee b \leftarrow {\sim}c;\ \ b \leftarrow {\sim}b;\ \ c \leftarrow {\sim}c\}.$$

Now a becomes false by the minimization of partial models, since the falsity of a does not affect the satisfiability of any rule. Thus the unique partial stable model of P is $M = \langle \emptyset, \{a\} \rangle$. Note that the reduction $P_M = \{a \vee b \leftarrow \mathbf{u};\ b \leftarrow \mathbf{u};\ c \leftarrow \mathbf{u}\}$. Then consider the translation

$$\begin{aligned}\mathrm{Tr}(P) = \{\ & a \vee b \leftarrow {\sim}c^\bullet; && b \leftarrow {\sim}b^\bullet; && c \leftarrow {\sim}c^\bullet; \\ & a^\bullet \vee b^\bullet \leftarrow {\sim}c; && b^\bullet \leftarrow {\sim}b; && c^\bullet \leftarrow {\sim}c; \\ & a^\bullet \leftarrow a; && b^\bullet \leftarrow b; && c^\bullet \leftarrow c\ \ \}\end{aligned}$$

The unique stable model of $\mathrm{Tr}(P)$ is $N = \{b^\bullet, c^\bullet\}$ which represents (by CE2 and CE3) the setting that b and c are undefined and a is false in M.

It is well-known that a disjunctive program P may not have any partial stable models. In such cases, the translation $\mathrm{Tr}(P)$ should not have stable models either, if the translation $\mathrm{Tr}(P)$ is to be faithful.

Example 3.3 *Consider a program*

$$P = \{a \vee b \vee c \leftarrow;\ \ a \leftarrow {\sim}b;\ \ b \leftarrow {\sim}c;\ \ c \leftarrow {\sim}a\}$$

and $\mathrm{Tr}(P) =$

$$\begin{aligned}\{\ & a \vee b \vee c \leftarrow; && a \leftarrow {\sim}b^\bullet; && b \leftarrow {\sim}c^\bullet; && c \leftarrow {\sim}a^\bullet; \\ & a^\bullet \vee b^\bullet \vee c^\bullet \leftarrow; && a^\bullet \leftarrow {\sim}b; && b^\bullet \leftarrow {\sim}c; && c^\bullet \leftarrow {\sim}a\ \} \\ & \cup\, C\end{aligned}$$

where $C = \{a^\bullet \leftarrow a;\ \ b^\bullet \leftarrow b;\ \ c^\bullet \leftarrow c\}$.

Consider a partial model $M = \langle \{a, b\}, \emptyset \rangle$ of P and a total model $N = \{a, a^\bullet, b, b^\bullet, c^\bullet\}$ of $\mathrm{Tr}(P)$ *that satisfy the equations CE1–CE4 in Definition 3.1. Now the reduced program P_M is*

$$\{a \vee b \vee c \leftarrow;\ \ a \leftarrow \mathbf{f};\ \ b \leftarrow \mathbf{u};\ \ c \leftarrow \mathbf{f}\}$$

and since $M' = \langle \{a, b\}, \{c\} \rangle < M$ is a partial model of P_M, M is not a partial stable model of P. On the other hand, the reduct

$$\mathrm{Tr}(P)^N = \{a \vee b \vee c \leftarrow;\ \ a^\bullet \vee b^\bullet \vee c^\bullet \leftarrow;\ \ b^\bullet \leftarrow\} \cup C.$$

But $N' = \{a, a^\bullet, b, b^\bullet\} \subset N$ is a model of $\mathrm{Tr}(P)^N$ so that N is not a stable model of $\mathrm{Tr}(P)$. The reader may analyze the other candidates in a similar fashion. It turns out that P does not have partial stable models. Nor does $\mathrm{Tr}(P)$ have stable models.

The main result of this section is established using the characterizations of disjunctive stable models in Theorems 2.4 and 2.5 (see the appendix for a proof).

Theorem 3.4 *Let M be a partial interpretation of a disjunctive program P and N a total interpretation of the translation $\mathrm{Tr}(P)$ such that CE1–CE4 are satisfied. Then M is a partial stable model of P iff N is a stable model of $\mathrm{Tr}(P)$.*

Let us yet address the possibility of using an inference engine for computing total stable models to answer queries concerning partial stable models. This is highly interesting, because there are already systems available for computing total stable models (Cholewiński et al. 1996, Leone et al. 1999, Simons 1999b) while partial stable models lack implementations. Here we must remind the reader that partial stable models can be used in different ways in order to evaluate queries. Typically two modes of reasoning are used: *certainty inference* and *possibility inference*. In the former approach, a query Q should be true in all (intended) models of P while Q should be true in some (intended) model of P in the latter approach. Moreover, maximal partial stable models (under set inclusion) are sometimes distinguished; this is how *regular models* and *preferred extensions* are obtained for normal programs (Dung 1995, Saccà and Zaniolo 1990, You and Yuan 1994). We are particularly interested in possibility inference where the maximality condition makes no difference: $M(Q) = \mathbf{t}$ for some partial stable model M of P iff $M'(Q) = \mathbf{t}$ for some maximal partial stable model M' of P.

We consider queries Q that are sets of literals over $\mathrm{Hb}(P)$ and queries are translated as follows: $\mathrm{Tr}(Q) = Q \cup Q^\bullet$. As a direct consequence of Theorem 3.4 and CE1, we obtain the following.

Corollary 3.5 *A query Q is true in a (maximal) partial stable model of P iff $\mathrm{Tr}(Q)$ is true in a stable model of $\mathrm{Tr}(P)$.*

What about using an inference engine for computing partial stable models to answer queries concerning stable models? A slight extension of the translation $\mathrm{Tr}(P)$ is needed for this purpose: let $\mathrm{Tr}_2(P)$ be $\mathrm{Tr}(P)$ augmented with a set of rules $\{f \leftarrow a^\bullet, {\sim}a \mid a \in \mathrm{Hb}(P)\}$ where $f \notin \mathrm{Hb}(P)$ is a new atom. The purpose of these additional rules is to detect partial stable models with remaining undefined atoms. A query Q is translated into $\mathrm{Tr}_2(Q) = Q \cup \{{\sim}f\}$.

Corollary 3.6 *A query Q is true in a stable model of P if and only if $\mathrm{Tr}_2(Q)$ is true in a partial stable model of $\mathrm{Tr}_2(P)$.*

4 UNFOLDING DISJUNCTIONS

In this section we develop a method for reducing the task of computing a (total) stable model of a disjunctive program to stable model computation for normal (disjunction-free) programs. Since the problem of deciding whether a disjunctive program has a stable model is Σ_2^p-complete (Eiter and Gottlob 1995) whereas the problem is NP-complete in the non-disjunctive case (Marek and Truszczyński 1991), the reduction cannot be computable in polynomial time unless the polynomial hierarchy collapses. Our idea is to use a straightforward generate and test approach.

The basic idea is that given a disjunctive program P we compute its stable models in two phases: (i) we *generate* model candidates and (ii) *test* candidates for stability until we find a suitable model. For generating model candidates we construct a normal program $\mathrm{Gen}(P)$ such that the stable models of $\mathrm{Gen}(P)$ give the candidate models. For testing a candidate model M we build another normal program $\mathrm{Test}(P, M)$ such that $\mathrm{Test}(P, M)$ has no stable models iff M is a stable model of the original disjunctive program P. Hence, given a procedure for computing stable models for normal programs all stable models of a disjunctive program P can be generated as follows: for each stable model M of $\mathrm{Gen}(P)$, decide whether $\mathrm{Test}(P, M)$ has a stable model and if this is not the case, output M as a stable model of P.

It is easy to construct a normal program $\mathrm{Gen}(P)$ for generating candidate models, e.g., by introducing for each atom $a \in \mathrm{Hb}(P)$, two rules $a \leftarrow {\sim}\hat{a}$; $\hat{a} \leftarrow {\sim}a$ where $\hat{a}$ is a new atom. These rules generate stable models corresponding to every subset of $\mathrm{Hb}(P)$. In order to prune this set of models to those with all rules in P satisfied, it is sufficient to include a rule

$$f \leftarrow {\sim}f, {\sim}a_1, \ldots, {\sim}a_k, b_1, \ldots, b_m, {\sim}c_1, \ldots, {\sim}c_n \quad (2)$$

for each rule of the form (1) in P where f is a new atom. As f cannot be in any stable model, the rule functions as an integrity constraint eliminating the models where each b_i is included, every c_j is excluded but no a_l is included.

In order to guarantee completeness, it is sufficient that for each stable model M of P there is a corresponding

model candidate which agrees with M w.r.t. $\mathrm{Hb}(P)$. It is clear that our first generating program satisfies this condition. However, for efficiency it is important to devise a generating program that has as few as possible (candidate) stable models provided that completeness is not lost. For given a disjunctive program P, a generating program $\mathrm{Gen}(P)$ that typically has far fewer stable models but still preserves completeness can be constructed as follows:

$$\begin{aligned}&\mathrm{Gen}(P) = \\ &\quad \{a \leftarrow \sim\hat{a}, B, \sim C \mid A \leftarrow B, \sim C \in P, a \in A\} \cup \\ &\quad \{\hat{a} \leftarrow \sim a \mid a \in \mathrm{Hb}(P)\} \cup \\ &\quad \{f \leftarrow \sim f, \sim A, B, \sim C \mid A \leftarrow B, \sim C \in P\}\end{aligned}$$

Proposition 4.1 *Let P be a disjunctive program. Then if M is a stable model of P, there is a stable model M' of $\mathrm{Gen}(P)$ with $M = M' \cap \mathrm{Hb}(P)$.*

Proof. Let M be a stable model of P and let $M' = M \cup \{\hat{a} \mid a \in \mathrm{Hb}(P) - M\}$. Now clearly $M = M' \cap \mathrm{Hb}(P)$. We show that M' coincides with the minimal model Min of $\mathrm{Gen}(P)^{M'}$, i.e., M' is a stable model of $\mathrm{Gen}(P)$. We prove this by showing that (i) M' is a model of $\mathrm{Gen}(P)^{M'}$ and that (ii) $M' \subseteq Min$ which together imply $M' = Min$.

Property (i) holds because M is a model of P and because $\hat{a} \leftarrow \in \mathrm{Gen}(P)^{M'}$ iff $a \in \mathrm{Hb}(P) - M$ and $a \leftarrow B \in \mathrm{Gen}(P)^{M'}$ iff $a \in M$ and $C \cap M = \emptyset$. In order to establish (ii) we note that (iii) for all $a \in \mathrm{Hb}(P)$, $a \in Min$ implies $a \in M$ because for each rule in $\mathrm{Gen}(P)^{M'}$ with such an atom a in the head, $a \in M$. Consider $A \leftarrow B \in P^M$. If the body B is true in Min, then by (iii) B is true in M and, hence, at least one $a_i \in A \cap M$. Then $a_i \leftarrow B \in \mathrm{Gen}(P)^{M'}$ and $a_i \in Min$. Hence, Min is a model of P^M which implies $M \subseteq Min$ and $M' \subseteq Min$. □

A (total) model candidate $M \subseteq \mathrm{Hb}(P)$ is a stable model of a program P if it is a minimal model of the GL-transform P^M of the program. This test can be reduced to an unsatisfiability problem in propositional logic using techniques presented in (Niemelä 1996): M is a minimal model of P^M iff

$$\begin{aligned}P^M \quad &\cup \quad \{\neg a \mid a \in \mathrm{Hb}(P) - M\} \\ &\cup \quad \{\neg b_1 \vee \cdots \vee \neg b_m\} \qquad (3)\end{aligned}$$

is unsatisfiable where $M = \{b_1, \ldots, b_m\}$ and the rules in P^M are seen as clauses. This problem can be solved by testing non-existence of stable models for a normal program $\mathrm{Test}(P, M)$ which is constructed for a disjunctive program P and a total interpretation $M \subseteq \mathrm{Hb}(P)$ as follows:

$$\begin{aligned}&\mathrm{Test}(P, M) = \\ &\quad \{a \leftarrow \sim\hat{a}, B \mid A \leftarrow B \in P^M, a \in A \cap M, B \subseteq M\} \cup \\ &\quad \{\hat{a} \leftarrow \sim a \mid a \in \mathrm{Hb}(P)\} \cup \\ &\quad \{f \leftarrow \sim f, \sim A, B \mid A \leftarrow B \in P^M, B \subseteq M\} \cup \\ &\quad \{f \leftarrow \sim f, M\}\end{aligned}$$

Proposition 4.2 *Let P be a disjunctive program and M its (total) model. Then M is a minimal model of P^M iff $\mathrm{Test}(P, M)$ has no stable model.*

Proof. Let $M \subseteq \mathrm{Hb}(P)$ be a total model of P.

($\Rightarrow$) Let M' be a stable model of $\mathrm{Test}(P, M)$. If $a \in \mathrm{Hb}(P) - M$, then there is no rule with a in the head in $\mathrm{Test}(P, M)$ and $a \notin M'$. Hence, $M' \cap \mathrm{Hb}(P) \subseteq M$. As $f \notin M'$ and $f \leftarrow M \in \mathrm{Test}(P, M)^{M'}$, there is some $a \in M$ such that $a \notin M' \cap \mathrm{Hb}(P)$. Consider $A \leftarrow B \in P^M$. Let $B \subseteq M' \cap \mathrm{Hb}(P) \subseteq M$ but suppose $A \cap M' \cap \mathrm{Hb}(P) = \emptyset$. Then $f \leftarrow B \in \mathrm{Test}(P, M)^{M'}$ and $f \in M'$, a contradiction. Hence, $M' \cap \mathrm{Hb}(P)$ is a model of P^M but $M' \cap \mathrm{Hb}(P) \subset M$ implying that M is not a minimal model of P^M.

($\Leftarrow$) Assume that M is not a minimal model of P^M. As M is a model of P^M, there is a minimal model $Min \subset M$ of P^M.

We show that $M' = Min \cup \{\hat{a} \mid a \in \mathrm{Hb}(P) - Min\}$ coincides with the minimal model Min' of $\mathrm{Test}(P, M)^{M'}$, i.e., M' a stable model of $\mathrm{Test}(P, M)$. Now $\mathrm{Test}(P, M)^{M'} =$

$$\begin{aligned}&\{a \leftarrow B \mid A \leftarrow B \in P^M, \\ &\qquad a \in A \cap M \cap Min, B \subseteq M\} \cup \\ &\{\hat{a} \leftarrow \mid a \in \mathrm{Hb}(P) - Min\} \cup \\ &\{f \leftarrow B \mid A \leftarrow B \in P^M, A \cap Min = \emptyset, B \subseteq M\} \cup \\ &\{f \leftarrow M\}\end{aligned}$$

It is easy to check that (i) M' is a model of $\mathrm{Test}(P, M)^{M'}$. Furthermore, (ii) $M' \subseteq Min'$ holds which can be established as follows. We notice that for all $a \in \mathrm{Hb}(P)$, $a \in Min'$ implies $a \in Min$. Consider $A \leftarrow B \in P^M$. If B is true in Min', then B is true in Min and, thus, $B \subseteq M$ and some $a \in A \cap M \cap Min$. Then $a \leftarrow B \in \mathrm{Test}(P, M)^{M'}$ and $a \in Min'$. Hence, Min' is a model of P^M which implies $Min \subseteq Min'$. Then $M' \subseteq Min'$ holds. Now (i) and (ii) imply $Min' = M'$. Hence, M' is a stable model of $\mathrm{Test}(P, M)$. □

Example 4.3 *Consider a disjunctive program P and*

its generator $\mathrm{Gen}(P)$*:*

$$P = \{a \vee b \leftarrow \sim c\}$$

$$\begin{aligned}\mathrm{Gen}(P) = \{\, & a \leftarrow \sim\hat{a}, \sim c;\ \ b \leftarrow \sim\hat{b}, \sim c;\\ & \hat{a} \leftarrow \sim a;\ \ \hat{b} \leftarrow \sim b;\ \ \hat{c} \leftarrow \sim c;\\ & f \leftarrow \sim f, \sim a, \sim b, \sim c\ \ \}\end{aligned}$$

For stable models $\{a, b, \hat{c}\}$ *and* $\{b, \hat{a}, \hat{c}\}$ *of* $\mathrm{Gen}(P)$ *the corresponding model candidates are* $M_1 = \{a, b, \hat{c}\} \cap \mathrm{Hb}(P) = \{a, b\}$ *and* $M_2 = \{b, \hat{a}, \hat{c}\} \cap \mathrm{Hb}(P) = \{b\}$ *and their test programs:*

$$\begin{aligned}\mathrm{Test}(P, M_1) = \{\, & a \leftarrow \sim\hat{a};\ \ b \leftarrow \sim\hat{b};\\ & \hat{a} \leftarrow \sim a;\ \ \hat{b} \leftarrow \sim b;\ \ \hat{c} \leftarrow \sim c;\\ & f \leftarrow \sim f, \sim a, \sim b;\ \ f \leftarrow \sim f, a, b\ \}\\ \mathrm{Test}(P, M_2) = \{\, & b \leftarrow \sim\hat{b};\\ & \hat{a} \leftarrow \sim a;\ \ \hat{b} \leftarrow \sim b;\ \ \hat{c} \leftarrow \sim c;\\ & f \leftarrow \sim f, \sim a, \sim b;\ \ f \leftarrow \sim f, b\ \}\end{aligned}$$

$\mathrm{Test}(P, M_1)$ *has a stable model (e.g.,* $\{a, \hat{b}, \hat{c}\}$*) indicating that* M_1 *is not a stable model of* P *but* $\mathrm{Test}(P, M_2)$ *has no stable models and, hence,* M_2 *is a stable model of* P*.*

The simple generate and test paradigm can be optimized by building model candidates gradually. This means that we start from the empty partial interpretation and extend the interpretation step by step. An interesting observation is that the technique for testing minimality can be used to rule out a partial model candidate of $\mathrm{Gen}(P)$ at any stage of the search and not just when a total model of the program P has been found. This can be done by treating a partial interpretation M as a total interpretation where undefined atoms are taken to be false and using the $\mathrm{Test}(P, M)$ program.

Proposition 4.4 *Let* P *be a disjunctive program and* M *a total interpretation. If* $\mathrm{Test}(P, M)$ *has a stable model, then there is no (total) stable model* M' *of* P *such that* $M \subseteq M'$*.*

Proof. Let $\mathrm{Test}(P, M)$ have a stable model. As shown in the proof of Proposition 4.2, then there is a model M'' of P^M with $M'' \subset M$. Consider any total interpretation M' such that $M \subseteq M'$ and M' is a model of $P^{M'}$. Now M' is not a minimal model of $P^{M'}$ as $P^{M'} \subseteq P^M$ and, hence, M'' is a model of $P^{M'}$ but $M'' \subset M \subseteq M'$. □

Notice that for a total interpretation M, Proposition 4.4 can only be used for eliminating stable models of P extending M. For guaranteeing the existence of a stable model of P, a total model of P needs to be found making Proposition 4.2 applicable.

5 EXPERIMENTS

We compare *dlv* (Leone et al. 1999), a state-of-the-art implementation of the stable model semantics for disjunctive logic programs, with an implementation of the generate and test approach (GnT) of the previous section. The implementation is based on *smodels* (Simons 1999b), a program that computes stable models of normal logic programs, and it is available (Simons 1999a).

We create two instances of *smodels*, one that generates the models and one that tests if they are minimal. Each time we find a model we check if it is minimal. If it is not, then we backtrack and perform the minimality test (Proposition 4.4) by viewing the resulting partial model as a total model. If the test fails again, then we repeat and backtrack until the test succeeds. Next, we resume the search for more models. The implementation of the procedure consists of less than 400 lines of code on top to the *smodels* system.

The two systems are tested on 3-SAT problems from the phase transition region. Given a propositional formula in conjunctive normal form whose clause to atom ratio is 4.258, we translate each clause $a_1 \vee \cdots \vee a_n \vee \neg b_1 \vee \cdots \vee \neg b_m$ into a rule

$$a_1 \vee \cdots \vee a_n \leftarrow b_1, \ldots, b_m.$$

In order to make the problem Σ_2^p-complete, we search for minimal models that contain some specified atoms (Eiter and Gottlob 1995). In particular, we create the rules

$$f \leftarrow \sim f, \sim c_i$$

for $i = 1, \ldots, \lfloor 2n/100 \rfloor$ and for random atoms c_i, where n is the number of atoms in the formula.

We generate random problems that have between 100 and 340 atoms. For each problem size we test fifty programs and measure the maximum, average, and minimum time it takes to decide whether a stable model exists. We have also tested the strategic companies problem from (Koch and Leone 1999), but we do not include the results as it turns out that the problem is almost trivial for both *dlv* and the GnT approach. All tests were run under Linux 2.2.12 on 450 MHz Pentium III computers with 256 MB of memory. The test results are shown in Figure 1.

For this set of benchmark problems, our GnT approach is very competitive when compared *dlv* except for two test problems (one having 270 atoms and the other 300 atoms) where the number of generated model candidates is orders of magnitude higher than in any of the other 49 test problems of the same size.

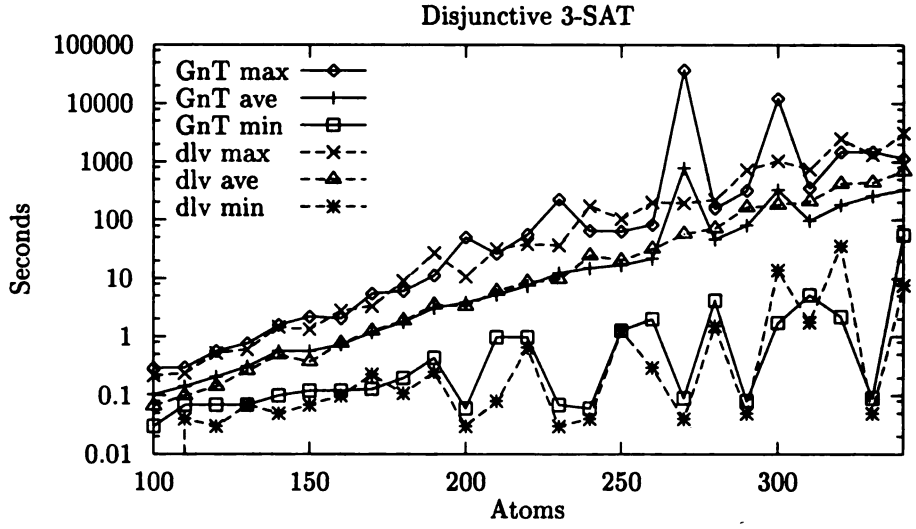


Figure 1: Experimental Results

6 CONCLUSIONS

The paper presents an approach to implementing partial and disjunctive stable models using an implementation of stable models for disjunction-free programs as the core inference engine. The approach is based on unfolding partiality and disjunctions from a logic program in two separate steps. In the first step partial stable models are captured by total stable models using a simple linear program transformation. Thus, reasoning tasks concerning partial models can be solved using an implementation of total models. This also sheds new light on the relationship between partial and total stable models by establishing a close correspondence. In the second step a generate and test approach is developed for computing total stable models of disjunctive programs using a core engine capable of computing stable models of normal programs. We have developed an implementation of the approach using *smodels* as the core engine. The extension is fairly simple consisting of a few hundred lines of code. The approach turns out to be competitive even against a state-of-the-art special purpose system for disjunctive programs. The efficiency of the approach comes partly from the fact that normal programs can capture essential properties of disjunctive stable models that help decrease the computational complexity of the generate and test phases in the approach. However, a major part of the success can be accounted for by the efficiency of the core engine. This suggests that more efforts should be spent in developing efficient core engines.

Acknowledgements

The work of the first, second and third authors has been funded by the Academy of Finland (Project 43963), that of the third author by the Helsinki Graduate School in Computer Science and Engineering. We thank Tommi Syrjänen for implementing the front-end for the *smodels* system.

References

Cholewiński, P., Marek, V. and Truszczyński, M. (1996). Default reasoning system DeReS, *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann Publishers, Cambridge, MA, USA, pp. 518–528.

Dimopoulos, Y., Nebel, B. and Koehler, J. (1997). Encoding planning problems in non-monotonic logic programs, *Proceedings of the Fourth European Conference on Planning*, Springer-Verlag, Toulouse, France, pp. 169–181.

Dung, P. (1995). An argumentation theoretic foundation for logic programming, *Journal of Logic Programming* **22**: 151–177.

Eiter, T. and Gottlob, G. (1995). On the computational cost of disjunctive logic programming: Propositional case, *Annals of Mathematics and Artificial Intelligence* **15**: 289–323.

Eiter, T., Leone, N. and Saccà, D. (1997). On the partial semantics for disjunctive deductive databases, *Annals of Mathematics and Artificial Intelligence* **17**(1/2): 59–96.

Eiter, T., Leone, N. and Saccà, D. (1998). Expressive power and complexity of partial models for disjunctive deductive databases, *Theoretical Computer Science* **206**(1/2): 181–218.

Eshghi, K. and Kowalski, R. (1989). Abduction compared with negation by failure, *Proceedings of the 6th International Conference on Logic Programming*, MIT Press, pp. 234–254.

Gelfond, M. and Lifschitz, V. (1988). The stable model semantics for logic programming, *Proceedings of the 5th International Conference on Logic Programming*, The MIT Press, Seattle, USA, pp. 1070–1080.

Heljanko, K. (1999). Using logic programs with stable model semantics to solve deadlock and reachability problems for 1-safe Petri nets, *Fundamenta Informaticae* **37**(3): 247–268.

Koch, C. and Leone, N. (1999). Stable model checking made easy, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, pp. 70–75.

Leone, N. et al. (1999). Dlv, release 1999-11-24, `http://www.dbai.tuwien.ac.at/proj/dlv/`. A Disjunctive Datalog System.

Leone, N., Rullo, L. and Scarcello, F. (1997). Disjunctive stable models: Unfounded sets, fixpoint semantics, and computation, *Information and Computation* **135**(2): 69–112.

Liu, X., Ramakrishnan, C. and Smolka, S. A. (1998). Fully local and efficient evaluation of alternating fixed points, *in* B. Steffen (ed.), *Proceedings of the 4th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer-Verlag, Lisbon, Portugal, pp. 5–19.

Marek, W. and Truszczyński, M. (1991). Autoepistemic logic, *Journal of the ACM* **38**: 588–619.

Marek, W. and Truszczyński, M. (1999). Stable models and an alternative logic programming paradigm, *The Logic Programming Paradigm: a 25-Year Perspective*, Springer-Verlag, pp. 375–398.

Niemelä, I. (1996). A tableau calculus for minimal model reasoning, *Proceedings of the Fifth Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, Springer-Verlag, Terrasini, Italy, pp. 278–294.

Niemelä, I. (1999). Logic programming with stable model semantics as a constraint programming paradigm, *Annals of Mathematics and Artificial Intelligence* **25**(3,4): 241–273.

Niemelä, I. and Simons, P. (1997). Smodels – an implementation of the stable model and well-founded semantics for normal logic programs, *Proceedings of the 4th International Conference on Logic Programming and Non-Monotonic Reasoning*, Springer-Verlag, Dagstuhl, Germany, pp. 420–429.

Przymusinski, T. (1990). Extended stable semantics for normal and disjunctive logic programs, *Proceedings of the 7th International Conference on Logic Programming*, MIT Press, pp. 459–477.

Saccà, D. and Zaniolo, C. (1990). Stable models and non-determinism in logic programs with negation, *Proceedings of the 9th ACM Symposium on Principles of Database Systems*, pp. 205–217.

Sagonas, K., Swift, T. and Warren, D. (1996). An abstract machine for computing the well-founded semantics, *in* M. Maher (ed.), *Proceedings of the Joint International Conference and Symposium on Logic Programming*, The MIT Press, Bonn, Germany, pp. 274–288.

Simons, P. (1999a). Disjunctive logic programming by generate and test, `http://www.tcs.hut.fi/Software/smodels/`. Available as examples/example4.cc in the smodels source distribution.

Simons, P. (1999b). Smodels version 2.25, `http://www.tcs.hut.fi/Software/smodels/`. A system for computing the stable models of logic programs.

You, J. and Yuan, L. (1994). A three-valued semantics for deductive databases and logic programs, *Journal of Computer and System Sciences* **49**: 334–361.

A APPENDIX: PROOF OF THEOREM 3.4

The main result of Section 3 is a theorem showing a one-to-one correspondence between the partial stable models M of a program P and the stable models N of its translation. Essentially, we prove that M and N form a pair that satisfies the correspondence equations CE1–CE4 given in Definition 3.1 by the intended representation of atoms and their truth values.

It is worth noting that although we assume ground programs, since our proofs do not rely on the assumption that the given program be finite, all of the results apply to the non-ground case as well.

The strategy to prove Theorem 3.4 is as follows: first, in two separate lemmas, we show the correspondence, in each direction, between unfounded sets for P and $\mathrm{Tr}(P)$ under M and N, respectively; these results will then be used to establish the required maximal unfoundedness, and foundedness in case of partial stable models. Before doing that we present two simple facts which can be verified easily.

Proposition A.1 *Let P be a program, N be a total interpretation. Then, X is an unfounded set for P w.r.t. N iff X is an unfounded set for P^N w.r.t. N.*

Proposition A.2 *Let M be a partial interpretation of a program P and N be a total interpretation of its translation $\mathrm{Tr}(P)$. Assume M and N satisfy the CEs. Then, M is a model of P iff N is a model of $\mathrm{Tr}(P)$.*

In the following, by abuse of terminology, an unfounded set X for $\mathrm{Tr}(P)$ is said to satisfy the consistency rules $\{a^\bullet \leftarrow a \mid a \in \mathrm{Hb}(P)\} \subseteq \mathrm{Tr}(P)$ iff $a^\bullet \in X$ implies $a \in X$. Since an unfounded set is about atoms that can be assumed false, this ensures that if the head of a consistency rule is false, the body must be false. An atom $a^\bullet$ is said to be *marked*, and an ordinary atom a is then said to be *unmarked*.

We also use the following notations: given a rule r, $H(r)$ refers to the set of head atoms in r, $B^+(r)$ refers to the set of positive body literals in r, $B^-(r)$ denotes the set of negative body literals in r, and $B(r)$ is simply $B^+(r) \cup B^-(r)$. Recall that in the translation each rule r: $A \leftarrow B, {\sim}C$ is translated into two rules $A \leftarrow B, {\sim}C^\bullet$ and $A^\bullet \leftarrow B^\bullet, {\sim}C$. The former is referred to as $r_{n'}$ (negative literals marked) and the latter as $r_{p'}$ (positive literals marked).

Lemma A.3 *Let P be a program, N be a total interpretation of the translated program $\mathrm{Tr}(P)$, and M be a partial interpretation of P satisfying the CEs. Then, for any unfounded set X_N for $\mathrm{Tr}(P)$ w.r.t. N satisfying the consistency rules, $X_M = \{a \mid a, a^\bullet \in X_N\}$ is an unfounded set for P w.r.t. M. In addition, if X_N is N-consistent then X_M is M-consistent.*

Proof. The additional claim on consistency is easy to show. Here we prove the main claim. Let $a \in X_M$. We prove that for any rule $r \in P$ such that $a \in H(r)$, one of the conditions for unfoundedness applies to r. We note that $B(r)$ is either false, or true, or undefined in M. Consider all three cases.

A. $B(r)$ is false in M. Then UF1 applies to r.

B. $B(r)$ is true in M. We show that either UF2 or UF3 applies to r. We first note that, by the definition of X_M and the fact that $a \in X_M$, we have $a^\bullet \in X_N$. Since X_N is unfounded w.r.t. N, and since in this case UF1 does not apply to $r_{p'}$ (by the CEs, that $B(r)$ is true in M implies that $B(r_{p'})$ is true in N) we know that either UF2 or UF3 applies to $r_{p'}$.

(i) *UF2 applies to $r_{p'}$.* Then, $\exists c^\bullet \in B(r_{p'})$ such that $c^\bullet \in X_N$. Since the rule $c^\bullet \leftarrow c$ is satisfied by X_N, we have $c \in X_N$. That $c, c^\bullet \in X_N$ implies $c \in X_M$. Thus UF2 applies to r.

(ii) *UF3 applies to $r_{p'}$.* Then, $\exists b^\bullet \in H(r_{p'})$ such that $b^\bullet$ is true in N and $b^\bullet \notin X_N$. That $b^\bullet$ is true in N implies that b is not false in M. By the definition of X_M, $b^\bullet \notin X_N$ implies $b \notin X_M$. Thus UF3 applies to r.

C. $B(r)$ is undefined in M. Then by the CEs, $B(r_{p'})$ is true in N. Since $a^\bullet \in X_N$ and $a^\bullet \in H(r_{p'})$, either UF2 or UF3 applies to $r_{p'}$. The proof follows from (i) and (ii) above. □

The condition that the consistency rules be satisfied in the above lemma is tight, as illustrated by the following example.

Example A.4 *Consider $P = \{a \vee b \leftarrow c_1, c_2\}$ and its translation*

$$\mathrm{Tr}(P) = \{\, a \vee b \leftarrow c_1, c_2;\quad a^\bullet \vee b^\bullet \leftarrow c_1{}^\bullet, c_2{}^\bullet;\quad a^\bullet \leftarrow a;\ b^\bullet \leftarrow b;\quad c_1{}^\bullet \leftarrow c_1;\ c_2{}^\bullet \leftarrow c_2 \,\}$$

Let $N = \{a, a^\bullet, b, b^\bullet, c_1, c_1{}^\bullet, c_2{}^\bullet\}$ be an interpretation for $\mathrm{Tr}(P)$. Then the corresponding interpretation for P is $M = \langle\{a, b, c_1\}, \emptyset\rangle$. Consider $X_N = \{a, a^\bullet, b, b^\bullet, c_1, c_2{}^\bullet\}$. It can be verified easily that X_N is unfounded for $\mathrm{Tr}(P)$ w.r.t. N. Also note that since $c_2{}^\bullet \in X_N$ and $c_2 \notin X_N$, X_N does not satisfy the consistency rule $c_2{}^\bullet \leftarrow c_2$. We can verify that the corresponding $X_M = \{a, b\}$ (as defined in the lemma) is

not unfounded for P w.r.t. M; for the only rule in P, its body is undefined thus UF1 does not apply; UF2 does not apply either since no atom in X_M that appears in the body of the rule; and finally UF3 fails to be applicable too since both a and b are in X_M.

Lemma A.5 *Let M be a partial model of a program P and N be a total interpretation of $\mathrm{Tr}(P)$ satisfying the CEs. Suppose X_M is a set of ordinary atoms, and $X_N = F \cup U$ where $F = \{a, a^\bullet \mid a \in X_M\}$, and U is any subset of $\{a \mid a \in N^{\mathbf{f}}, a^\bullet \in N^{\mathbf{t}}\}$. Then, if X_M is an M-consistent unfounded set for P w.r.t. M, then X_N is an unfounded set for $\mathrm{Tr}(P)$ w.r.t. N.*

Proof. Any atom in X_N is either marked or unmarked. We prove the theorem by considering these two cases.

I. $a^\bullet \in X_N$. First, for the consistency rule $a^\bullet \leftarrow a \in \mathrm{Tr}(P)$, by the definition of X_N, it is clear that if $a^\bullet \in X_N$ then $a \in X_N$, so UF2 applies. For any other rule $r_{p'} \in \mathrm{Tr}(P)$ such that $a^\bullet \in H(r_{p'})$, we prove that one of the three conditions for unfoundedness applies to $r_{p'}$. Since N is a total interpretation, we have that $B(r_{p'})$ is either false in N, in which case UF1 applies to $r_{p'}$, or true in N in which case $B(r)$ is true or undefined in M. Since $a \in X_M$ and $a \in H(r)$, and UF1 does not apply to r, we only need to consider UF2 and UF3. If UF2 applies to r, $\exists c \in B(r)$ such that $c \in X_M$. It follows, by the definition of X_N, that $c^\bullet \in X_N$. Hence UF2 applies to $r_{p'}$. If UF3 applies to r, $\exists b \in H(r)$ such that b is not false in M and $b \notin X_M$. By the CEs, and by the fact that b is not false in M, we know $b^\bullet$ is true in N. Further, by the definition of X_N, $b \notin X_M$ implies $b^\bullet \notin X_N$. Hence UF3 applies to $r_{p'}$.

II. $a \in X_N$. For any $r_{n'} \in \mathrm{Tr}(P)$ such that $a \in H(r_{n'})$, $B(r_{n'})$ is either false in N, in which case UF1 applies to $r_{n'}$, or true in N, in which case, by Proposition A.2 we know that M being a model of P implies N is a model of $\mathrm{Tr}(P)$, hence $\exists b \in H(r_{n'})$ such that b is true in N. Two cases arise: there exists one such b that $b \notin X_N$, or for all such b, $b \in X_N$. In the first case, UF3 applies to $r_{n'}$. The proof is completed by showing that the latter case leads to a contradiction.

In this latter case, for each such b, from $b \in X_N$ we get $b \in F$ or $b \in U$. If $b \in F$ then $b, b^\bullet \in F$, hence $b \in X_M$. But b being true in N implies that b is true in M, and by $b \in X_M$ we get $M^{\mathbf{t}} \cap X_M \neq \emptyset$. This contradicts the assumption that X_M is M-consistent. Thus, for any such b, it must be the case that $b \in U$. By the definition of U this means that b must be false in N. A contradiction. This completes the proof. □

We note that M-consistency of X_M is also a necessary condition for the correspondence to hold.

Example A.6 *Consider the following program*

$$P = \{a \vee b \leftarrow;\ a \leftarrow \sim a\}$$

and a partial model $M = \langle\{b\}, \emptyset\rangle$ of P. It can be checked easily that $X_M = \{b\}$ is an unfounded set for P w.r.t. M (UF3 applies to the only rule in which b appears in the head). But X_M is not M-consistent. Now consider

$$\mathrm{Tr}(P) = \{\, a \vee b \leftarrow;\quad a^\bullet \vee b^\bullet \leftarrow;\quad a \leftarrow \sim a^\bullet;\quad a^\bullet \leftarrow \sim a;\quad a^\bullet \leftarrow a;\quad b^\bullet \leftarrow b \,\}$$

The total interpretation corresponding to M above is $N = \{b^\bullet, b, a^\bullet\}$. However, $X_N = \{b, b^\bullet, a\}$ is not unfounded for $\mathrm{Tr}(P)$ w.r.t. N, since for $b \in X_N$ and the first rule in $\mathrm{Tr}(P)$, none of the unfoundedness conditions applies.

Let us establish Theorem 3.4 in two separate theorems. Let M be a partial interpretation of a disjunctive program P and N a total interpretation of $\mathrm{Tr}(P)$ such that CE1–CE4 are satisfied.

Theorem A.7 *If N is a stable model of $\mathrm{Tr}(P)$, then M is a partial stable model of P.*

Proof. We prove that $M^{\mathbf{t}}$ is a minimal total model of P^M, and $M^{\mathbf{f}}$ is a maximal M-consistent unfounded set for P w.r.t. M.

I. Prove that $M^{\mathbf{t}}$ is a minimal total model of P^M. First observe that $P^M \subset \mathrm{Tr}(P)^N$. Let us partition $\mathrm{Tr}(P)^N$ into two disjoint sets $\Phi_1 = P^M$ and $\Phi_2 = \mathrm{Tr}(P)^N - P^M$. Now suppose $M^{\mathbf{t}}$ is not a minimal total model of P^M. This leads to one of the two possibilities

- $M^{\mathbf{t}}$ is not a total model of P^M. Since $P^M \subset \mathrm{Tr}(P)^N$, by the CEs, it follows that N is not a model of $\mathrm{Tr}(P)^N$, contradicting the assumption that N is a stable model of $\mathrm{Tr}(P)$.
- $M^{\mathbf{t}}$ is a model of P^M but not minimal. Then, there exists $S \subset M^{\mathbf{t}}$ which is a (total) model of P^M. We construct a total interpretation: $I = S \cup \{a^\bullet \mid a^\bullet \in N\}$. By $S \subset M^{\mathbf{t}}$, and by the CEs, we have $I \subset N^{\mathbf{t}}$ (i.e., only some marked atoms in $N^{\mathbf{t}}$ are not in I). Since $\Phi_1 = P^M$ and S is a model of P^M, every rule in Φ_1 is satisfied by I. Since the marked atoms in I are precisely those in N, any rule $r \in \Phi_2$ is also satisfied in I. Hence I is a model of $\mathrm{Tr}(P)^N$. It follows from $I \subset N^{\mathbf{t}}$ that $M^{\mathbf{t}}$ is not a minimal model of $\mathrm{Tr}(P)^N$, therefore not a stable model of P. A contradiction.

II. Prove that $M^{\mathbf{f}}$ is a maximal M-consistent unfounded set for P w.r.t. M. Since N is a model of

$\mathrm{Tr}(P)$, $N^{\mathbf{f}}$ satisfies $\{a^{\bullet} \leftarrow a \mid a \in \mathrm{Hb}(P)\}$. $N^{\mathbf{f}}$ is also M-consistent since $N^{\mathbf{f}} \cap M^{\mathbf{t}} = \emptyset$. It follows from Lemma A.3 that $M^{\mathbf{f}}$ is an M-consistent unfounded set for P w.r.t. M. We show its maximality by proving the contrapositive: if $M^{\mathbf{f}}$ is not a maximal M-consistent unfounded set for P w.r.t. M, then N cannot be a stable model of P.

Assume $M^{\mathbf{f}}$ is not maximal. Then there is an M-consistent unfounded set X for P w.r.t. M such that $X \supset M^{\mathbf{f}}$. That is, there is a such that $a \in X$ but $a \notin M^{\mathbf{f}}$. That $a \notin M^{\mathbf{f}}$ means $a \in M^{\mathbf{t}}$ or $a \in M^{\mathbf{u}}$. In both cases, by the CEs, $a^{\bullet} \in N^{\mathbf{t}}$, i.e., $a^{\bullet} \notin N^{\mathbf{f}}$. We construct a set $X' = \{a, a^{\bullet} \mid a \in X\}$. According to Lemma A.5, that X is an M-consistent unfounded set for P w.r.t. M implies that X' is an unfounded set for $\mathrm{Tr}(P)$ w.r.t N. However, $a \in X$ implies $a^{\bullet} \in X'$ but $a^{\bullet} \notin N^{\mathbf{f}}$. Thus $a^{\bullet} \in N^{\mathbf{t}}$ indicating that N is not unfounded free for $\mathrm{Tr}(P)$. Consequently, N is not a stable model of $\mathrm{Tr}(P)$ by Theorem 2.4. □

Theorem A.8 *If M is a partial stable model of P, then N is a stable model of* $\mathrm{Tr}(P)$.

Proof. Since M is a partial stable model of P, we know by Theorem 2.5 that (i) $M^{\mathbf{t}}$ is a minimal model of P^M and (ii) $M^{\mathbf{f}}$ is a maximal M-consistent unfounded set for P w.r.t. M. We note again that it follows by the CEs that $\mathrm{Tr}(P)^N = P^M \cup P_m$ where P_m is the union of the sets of rules $\{a^{\bullet} \leftarrow a \mid a \in \mathrm{Hb}(P)\}$ and

$$\{H(r_{P'}) \leftarrow B^+(r_{P'}) \mid r_{P'} \in \mathrm{Tr}(P), M(B^-(r_{P'})) = \mathbf{t}\}.$$

That is, $\mathrm{Tr}(P)^N$ can be partitioned into two disjoint sets of rules where P^M consists of the rules that only involve unmarked atoms and P_m consists of the rest of the rules with marked atoms in their heads. Similarly, N can be partitioned into two disjoint subsets, N_m and N_u, where N_m consists of the marked atoms in N and N_u the unmarked atoms in N.

Note that $M^{\mathbf{t}}$ is the total model obtained from M by making the undefined false. This is precisely N_u (which contains those atoms that are either false or undefined in M). Then, when $M^{\mathbf{t}}$ is a minimal model of P^M, it is precisely the case that N_u is a minimal model of the subset P^M of $\mathrm{Tr}(P)^N$.

Assume that N is not a stable model of $\mathrm{Tr}(P)$. Equivalently, N is not unfounded free for $\mathrm{Tr}(P)$. Then, there is an unfounded set X for $\mathrm{Tr}(P)$ w.r.t. N such that $X \cap N^{\mathbf{t}} \neq \emptyset$. Without loss of generality, assume X is maximal. From Proposition A.1, we know that X is a maximal unfounded set for $\mathrm{Tr}(P)^N$ w.r.t. N. Since N_u is a minimal model of the subset P^M of $\mathrm{Tr}(P)^N$, and since P^M is independent of P_m, it is clear that ${N_u}^{\mathbf{f}} \subseteq X$. In addition, any atom that is in $X \cap N^{\mathbf{t}}$ must be marked. We show that this implies that $M^{\mathbf{f}}$ cannot be a maximal M-consistent unfounded set for P w.r.t. M, generating a contradiction.

Let $a^{\bullet} \in X \cap N^{\mathbf{t}}$. We define a set

$$S_X = \{a, a^{\bullet} \mid a^{\bullet} \in N^{\mathbf{f}}\} \cup \{a \mid a^{\bullet} \in X \text{ and } a \in M^{\mathbf{u}}\}$$

Note that $a \in M^{\mathbf{u}}$ implies $a^{\bullet} \in N^{\mathbf{t}}$. Further, one can verify easily that $a^{\bullet} \in X$ implies $a \in X$ (as X is unfounded, for $a^{\bullet} \leftarrow a \in \mathrm{Tr}(P)$, if $a \in N^{\mathbf{f}}$, by ${N_u}^{\mathbf{f}} \subseteq X$, we get $a \in X$; otherwise the application of UF2 requires $a \in X$). Thus, if $a \in S_X$ then $a \in M^{\mathbf{f}}$ or $a \in M^{\mathbf{u}}$, i.e., $S_X \cap M^{\mathbf{t}} = \emptyset$, hence S_X is consistent with M. From the assumption $\exists a^{\bullet} \in X \cap N^{\mathbf{t}}$, we have $M^{\mathbf{f}} \subset S_X$. By Lemma A.3, that X is an unfounded set for $\mathrm{Tr}(P)$ w.r.t. N implies that that S_X is an unfounded set for P w.r.t. M. In addition, S_X is M-consistent. However, from $M^{\mathbf{f}} \subset S_X$ we conclude that $M^{\mathbf{f}}$ is not a maximal M-consistent unfounded set for P w.r.t. M.

Thus N must be a stable model of $\mathrm{Tr}(P)$. □

Learning and Decision Making

Relational Representations that Facilitate Learning

Chad Cumby **Dan Roth**

Department of Computer Science
University of Illinois at Urbana-Champaign
{cumby,danr}@uiuc.edu

Abstract

Given a collection of objects in the world, along with some relations that hold among them, a fundamental problem is how to learn definitions of some relations and concepts of interest in terms of the given relations. These definitions might be quite complex and, inevitably, might require the use of quantified expressions. Attempts to use first order languages for these purposes are hampered by the fact that relational inference is intractable and, consequently, so is the problem of learning relational definitions.

This work develops an expressive relational representation language that allows the use of propositional learning algorithms when learning relational definitions.

The representation serves as an intermediate level between a raw description of observations in the world and a propositional learning system that attempts to learn definitions for concepts and relations. It allows for hierarchical composition of relational expressions that can be evaluated efficiently on the observations and thus supports learning complex definitions by learning simple functions of the intermediate representations. The approach is illustrated using examples from natural language and visual processing.

1 Introduction

Given a collection of objects in the world, along with some relations that hold among them, a fundamental problem is how to learn definitions for some relations or concepts of interest in terms of the given relations. This situation arises in a variety of AI problems such as natural language understanding related tasks, visual interpretation and planning. Examples include the problem of identifying noun or subject-verb phrases in a sentence in terms of the lower level information provided in the sentence, detecting faces in an image (phrased as "finding a definition for a *face* in terms of the information provided in the image"), or defining a policy that maps states and goals to actions in a planning situation.

One of the main challenges in these tasks is that they are knowledge intensive; a lot of information about the world and the domain (language, three dimensional world, faces, etc.) may be required to make a decision with respect to a given input. Consequently, a computational solution seems to require a significant learning component; learning is necessary both for the large scale knowledge acquisition required and in order to ensure robust behavior – given that a decision needs to integrate a large number of information sources evaluated on a naturally occurring input such as a sentence or an image. This realization is a prime reason for the gradual but dramatic shift in paradigm in AI. Recent works on natural language and visual processing inferences emphasize statistical learning methods and shy away from the previously dominant knowledge based paradigm; an approach which suggested [15] that decisions of this sort are to be inferred from knowledge that is programmed into the system in some uniform well defined language.

Statistical learning methods, which have been quite successful for low level identification tasks, have significant drawbacks as a comprehensive solution for higher level tasks that may require manipulating more expressive representations. Expressing complex concepts, we believe, will require representing relations over the input. In terms of these intermediate representations (e.g., the concepts of "part of speech" or "proper noun" for phrases, the concept of an "edge"

for vision processing, or that of "eye locations" for face recognition) the complex concepts will have simpler representations and learning it might be more manageable. Representing the intermediate notions, in turn, may rely on external information sources or be acquired in different ways, possibly unrelated to the current task, and might require some sort of inference.

We believe that the study of computational processes that interact with raw data and acquire the ability to make knowledge intensive inferences would benefit if learning, knowledge representation and inference processes could be studied within a unified framework. This research continues an effort to develop an integrated framework for the study of learning, knowledge representation and reasoning [10, 11, 24] and investigates a crucial stage in the computational process, studied in the context of a concrete collection of large scale problems.

We are interested in learning a definition for some relations or concepts of interest, a definition which can be evaluated efficiently given an instance in the domain. For this purpose we study intermediate knowledge representations, between a raw description of observations in the world and a learning system that attempts to learn the target definition. A search for representations of this sort need to address, as a minimum, the following:

(i) Expressivity: A target definition should have a simple functional form (e.g., a rule, a linear function), and thus be learnable, in terms of the intermediate representation. It might be too complex and therefore unlearnable in terms of the raw input. The use of quantified representations is important in this respect.

(ii) Evaluating the intermediate representation on the raw input should be done efficiently. This requires an efficient solution to the *subsumption* problem: given an input instance and a quantified intermediate representation, decide whether the quantified expression is satisfied by the input instance.

(iii) Uniformity and well defined semantics. An intermediate knowledge representation might be acquired in a variety of ways. Fragments of it may be learned, programmed into the system, or inferred. These should be abstracted away to allow its use by a learning system in a uniform way.

(iv) Finally, the intermediate representation, which serves as the input representation for the learning stage should be made available to it in a way that facilitates learning.

For knowledge intensive tasks, the number of potential information sources affecting each decision may be huge (e.g, any English word may be in the input sentence) – although only a small subset of these may be relevant to a decision. We therefore interpret the last requirement in a specific way – the intermediate representation should be available to propositional learning algorithms. The reason is that no other computationally viable approach can learn definitions that are non-trivial in size and structure, and do it in the presence of thousands of information sources or more.

This paper develops a intermediate representation language that addresses the above issues. After describing the learning scenario, the context in which the intermediate representations are to be used (Section 2) and the representation language - a restricted first order language (Section 3), Section 4 introduces the key notion of a *relation-generation function* (RGF). RGFs can be viewed as defining "types" of relations that may be of interest as intermediate representations. Given a raw description of observations in the worlds, relations of these "types" that actually *occur* in the input are generated and used as input "features" to the learning stage. The relations are elements in the representation language defined in Section 3 and, as we show, map efficiently to an input of propositional learning algorithms.

The simplest form of an RGF is a *sensor*, which is used to abstract the interaction with naturally occurring data. Sensors extract the primitive information available on the input instance. A relational calculus defined allows the composition of sensors into more expressive RGFs. Section 5 formalizes the notion of structural information. E.g., the structural information in an instance may simply describe the linear structure of a sentence or the (bottom-up)×(left-right) structure of an image. RGFs utilize this information to efficiently generate expressive representations. For example, a construct of a "conjunction of consecutive elements" may give rise to an *edge* if applied to an appropriate "thresholded intensity" sensor; a structural construct applied to "adjacency" may give rise to recursive notions like "above", or "before". Computational issues involved in evaluating RGFs and generating the input for learning are also discussed.

The main contribution of this work is in providing a way to use expressive representations in the context of learning. Specifically, it

- defines an expressive language so that further learning stages can view it propositionally,
- formalizes an interface with naturally occurring

data,

- introduces the notion of RGFs which allows a for non-explicit, data-driven generation of relations, and
- formalizes restrictions which allow efficient evaluation.

The approach described here has been implemented and used in several large scale problems [17, 21, 22]. In Section 6 we briefly discus implementation issues and a detailed example from the natural language domain that exhibits some of the notions introduced.

2 Learning Expressive Representations

This section describes the learning scenario at the basis of this work to provide the context in which relation-generation-functions and the relational representations they produced are being used. We would like to learn programs of the form[1]

$$subject(x) \doteq f(after(x, verb), before(x, determiner), noun(x), ..)$$

so that when the program is evaluated on a given sentence its consequent has truth value "true" iff x is bound to a *subject* in the given sentence. The function f can be, in principle, any efficiently evaluatable Boolean function. In particular, if f is a conjunction, this program is a single clause logic program.

We present this work in the context of a *supervised learning* paradigm, although nothing in the rest of the paper is restricted to this learning protocol. The main concern would be the representation of a domain instance, its mapping into a convenient representation for the learning procedure and, consequently, the representation of the learned program. The input to the learning algorithm is given in terms of an input sentence S along with an atom p that describes a property that holds in the sentence (a "label"); we would like to learn a definition (program) for this property in terms of the input sentence.

Inductive logic programming (ILP) [16, 3] is a natural paradigm for this learning problem. Computational limitations, however, render this approach inadequate for large scale knowledge intensive problems, like natural language inferences in a real world context. The limiting computational issues include both learnability [5, 4] (see [12] for a discussion) and the problem of *subsumption*[2] – deciding whether the premises of a rule cover a domain instance.

Two additional issues motivate the computational approach presented here. First, the need to learn in very large domains or in domains in which not all elements are known in advance; thus, it is impossible, or impractical, for a learning approach to write explicitly, in advance, all possible "features". Second, for knowledge intensive tasks, the number of potential information sources that may affect each decision is very large but, typically, only a small number of them is actually relevant to a decision. A realistic learning approach needs therefore to be feature efficient [13] in that its complexity depends on the number of relevant features and not the global number of those in the domain.

2.1 The Learning Approach

The learning approach that serves as context to this work is a propositional learning approach in an infinite attribute domain [1], such as the one presented in [19, 2]. The learning procedure in this case learns a mapping $h : \{0,1\}^\infty \rightarrow \{0,1\}$ (or another discrete range). As input, the algorithm receives labeled instances $< (x, l) >$, where an instance $x \in \{0,1\}^\infty$ is presented as a list of all the *active* (of value 1) features in it. We typically assume that the learning scenario is *on-line*, although this is not required for the current setting, and do not assume that labels l are explicitly given; these can simply be designated features in the representation of $x \in \{0,1\}^\infty$ (as done, for example, in the SNoW learning architecture [2]).

In the learning scenario suggested here, an instance $x \in \{0,1\}^\infty$ indicates a collection of formulae in the relational language $\mathcal{R}$ (Section 3) that have a truth value "true" in the domain instance. Given a domain instance (e.g., a sentence) a set of pre-existing relation generation functions (RGFs, Section 4) are evaluated on it and generate a collection of formulae that are *active* (have truth value "true") in this sentence. The names of these formulae are the identifiers of the features presented to the propositional learning procedure.

Basic RGFs may be viewed themselves as programs of type listed above for *subject(x)* (with some restrictions), but for the purpose of this exposition the source of the RGF is irrelevant; they can be computed explicitly, learned, or inferred. The paradigm presented here

[1] The example is meant only to illustrate the ideas and may not be accurate. Indeed, one reason learning is crucial in this domain is that it is hard to come up with concise rules that perform well.

[2] Subsumption is NP-complete [8] and its cost may be practically implausible in complex domains [23].

is made possible due to:

- The restricted formulae generated by the RGFs (Sec. 3, [12]) allow the efficient mapping of quantified expressions describing a given instance to a propositional instance.
- The consistency of the RGFs – identifiers (formulae) for properties satisfied by new instances are generated consistently thus enabling generalization.

Finally, the success of this paradigm depends on that (i) RGFs are given the flexibility to generate a large number of formulae, which are active in observed instance, but many of which may not be relevant to the decision of interest and (ii) a feature efficient learning algorithm receiving this input can tolerate this without a significant additional cost.

Programs learned by the propositional learning algorithms can then be translated to quantified expressions, by substituting for each proposition its corresponding quantified formula. While the basic formulae are not as expressive as in the ILP paradigm, the structure of the rules learned is, in general, more expressive than those used in ILP, since the function f in the *subject* definition above need not be a conjunction[3]. The learning scenario studied here is that of learning one "rule" f at a time. Chaining of these programs [17] will not be discussed here.

3 The Relational Language $\mathcal{R}$

The relational language $\mathcal{R}$ is a restricted first order language. The *alphabet* consists of (i) variables, (ii) constants, (iii) predicate symbols, (iv) quantifiers and (v) connectives. (ii) and (iii) vary from alphabet to alphabet while (i), (iv) and (v) are the same for every alphabet. Formulae in $\mathcal{R}$ are defined to be restricted first order language formulae in which there is only a single predicate in the scope of each variable.

Definition 3.1 *An* atomic formula *is defined inductively as follows:*

1. *A* term *is either a variable or a constant.*
2. *Let p be a k-ary predicate, $t_1, \ldots, t_k$ terms. Then $p(t_1, \ldots, t_k)$ is an atomic formula.*
3. *Let F be an atomic formula, z a variable. Then $(\forall zF)$ and $(\exists zF)$ are atomic formulae.*

[3]The paradigm has been applied with f as a linear function [17, 12] although others can be used.

Definition 3.2 *A* formula *is defined inductively as follows:*

1. *An* atomic formula *is a formula.*
2. *If F and G are formulae, then so are $(\sim F), (F \wedge G), (F \vee G)$.*

The relational language given by the alphabet consists of the set of all formulas constructed from the symbols of the alphabet. We call a variable-less atomic formula a *proposition* and a quantified atomic formula, a *quantified proposition* [12]. The informal semantics of the quantifiers and connectives is as usual.

Example 3.1 *An example of a bound atomic formula in $\mathcal{R}$ in an NLP scenario could be a relation word(dog), which evaluates to true if there exists a word "dog"" in the input instance. An unbound atomic formula could be a relation object(x), which would evaluate to true if a word exists in the input instance which is an object in it. The truth value of this formula is to be determined by a RGF, based on information given in the input instance (Section 4).*

For formulae in $\mathcal{R}$, the *scope* of a quantifier is always the unique predicate that occurs with it in the atomic formula.

Definition 3.3 *Let $f : \{0,1\}^n \rightarrow \{0,1\}$ be a Boolean function of n variables, and let $F_1, F_2, \ldots F_n$ be formulae in $\mathcal{R}$. A* clause *is a formula of the form $f(F_1, F_2, \ldots F_n)$.*

This can be used, in particular, to define disjunctive, conjunctive and implication clauses.

3.1 Interpretation

$\mathcal{R}$ is used as a language for representing knowledge with respect to a domain.

Definition 3.4 *A domain $\mathcal{D}$ for the language $\mathcal{R}$ is a collection D of elements along with*

(i) *For each constant in $\mathcal{R}$, an assignment of an element in D.*

(ii) *For each k-ary predicate in $\mathcal{R}$, the assignment of a mapping from D^k to $\{0,1\}$ ($\{true, false\}$).*

Relational variables in $\mathcal{R}$ receive their "truth values" in a data driven way, with respect to an observation.

Definition 3.5 *An* instance *is an interpretation[14] which lists a set of domain elements and the truth values of all instantiations of the predicates on them.*

Given an instance x, a formula F in $\mathcal{R}$ is given a unique truth value, *the value of F on x*, defined inductively using the truth values of the predicates in F, and the semantics of the connectives. Since for formulae in $\mathcal{R}$ the scope of a quantifier is always the unique predicate that occurs with it in the atomic formula, we have:

Proposition 3.1 *Let F be a formula in $\mathcal{R}$, x an instance, and let t_p be the time to evaluate the truth value of an atom p in F. Then, the value of F on x can be evaluated in time $\sum_{p \in F} t_p$.*

That is, F is evaluated simply by evaluating each of its atoms (ground or quantified) separately. This holds, similarly, for the following version of subsumption for formulae in $\mathcal{R}$.

Proposition 3.2 (subsumption) *Let x be an instance and let $f : \{0,1\}^n \rightarrow \{0,1\}$ be a Boolean function of n variables that can be evaluated in time t_f. Then the value of the clause $f(F_1, \ldots F_n)$ on x can be evaluated in time $t_f + \sum_F t_F$, where the sum is over all n formulae that are the arguments of f.*

4 Relation Generation Functions

Definition 4.1 *A formula in $\mathcal{R}$ maps an instance x to its truth value in x. A formula is* active *in x if it has truth value* true *in this instance. We denote by X the set of all instances, the* instance space. *A formula $F \in \mathcal{R}$ is thus a* relation *over X, $F : X \rightarrow \{0,1\}$.*

Example 4.1 (NLP) *An instance x could be an unordered collection of words: he, ball, the kick, would. Some active relations on this instance are word(he), word(ball), and tag(DET).*

Given an instance, we would like to know what are the relations (formulae) that are *active* in it. We would like to do that, though, without the need to write down explicitly all possible formulae in the domain. This is important, in particular, over infinite domains or in on-line situations where the domain elements are not known in advance, and therefore it is simply impossible to write down all possible formulae. An efficient way to do that is given by the construct of *relation generating functions*. As will be clear later, this notion will also allow us to significantly extend the language of formulae by exploiting properties of the domain.

Definition 4.2 *Let $\mathcal{X}$ be an enumerable collection of relations on X. A relation generation function (RGF) is a mapping $G : X \rightarrow 2^{\mathcal{X}}$ that maps $x \in X$ to a set of all elements in $\mathcal{X}$ that satisfy $\chi(x) = 1$. If there is no $\chi \in \mathcal{X}$ for which $\chi(x) = 1$, $G(x) = \phi$.*

RGFs can be thought of as a way to define "kinds" of formulae, or to parameterize over a large space of formulae. Only when an instance x is presented, a concrete formulae (or a collection of) is generated. An RGF can be thought of as having its own range $\mathcal{X}$ of relations.

4.1 Relational Calculus

The family of relation generation functions for $\mathcal{R}$ are RGFs whose output are formulae in $\mathcal{R}$. Those are defined inductively, just like the definition of the language $\mathcal{R}$.

The relational calculus is a calculus of symbols that allows one to inductively compose relation generation functions. The alphabet for this calculus consists of (i) basic RGFs, called *sensors* and (ii) a set of connectives. While the connectives are the same for every alphabet the *sensors* vary from domain to domain. A sensor is a way to encode basic information one can extract from an instance. It can also be used as a uniform way to incorporate external knowledge sources that aid in extracting information from an instance.

Definition 4.3 *A* sensor *is a relation generation function that maps an instance x into a set of atomic formulae in $\mathcal{R}$. When evaluated on an instance x a sensor s outputs all atomic formulae in its range which are* active.

Example 4.2 (NLP) *Given the instance x from EX. 4.1, two reasonable sensors to be given may be a sensor that observes the "word" relations and one for "tag" relations. The first could output the grounded relations: word(he), word(would), etc., and the existential relation word(z). The tag sensor would need to be able to extract part-of-speech information from the instance to generate relations over it. In this case it could generate the grounded relations: tag(DET), tag(V), tag(MOD), etc. and the existential relation tag(z). An ISA sensor could be a sensor which returns a semantic class for each word (e.g., for the word "kick", it might return the relation ISA(action)). It could be implemented, say, by accessing an external knowledge source.*

Example 4.3 (VP) *Interesting sensors for visual applications could include an (intensity > n) RGF, which could generate relations such as I>50(pixel5).*

Several mechanisms are used in the relational calculus to define the operations of RGFs. The following two restrict the range and the domain of an RGF, respectively.

Definition 4.4 *Let C be a set of formulae. A* conditioning *operation $|C$ on an RGF r restricts r to output only formulae in C.*

Example 4.4 *There are several notations used to define commonly used subsets of formulae. For example, for a sensor s, the notation $s_{|z}$ ($s_{|a}$, resp.) restricts that range of s to include only the corresponding existential atomic formulae (ground atomic formulae, resp.)*

Definition 4.5 *Let E be a set of elements in the domain D. An RGF r is* focused *on E if, given an instance x it generates only formulae in its range that are active in x due to elements in E. The focused RGF is denoted $r[E]$.*

Definition 4.6 *The operation of a* relation generation function *(RGF) for $\mathcal{R}$ is defined inductively as follows:*

1. *When evaluated on an instance x the sensor s outputs all* active *atomic formulae in its range.*
2. *If s and r are RGFs for $\mathcal{R}$, then so are $(\neg s), (s\&r), (s|r)$.*
   - i *The formulae in the output of $(\neg s)$ are active formulae of the form $\{\sim F\}$, where F is in the range of s (evaluated on x).*
   - ii *The formulae in the output of $(s\&r)$ are active formulae of the form $F \wedge G$, where F is in the range of s and G is in the range of r (evaluated on x).*
   - iii *The formulae in the output of $(s|r)$ are active formulae of the form $F \vee G$, where F is in the range of $s_{|C}$ and G is in the range of $r_{|C'}$, and C, (C', resp.) are formulas in $s[D]$ ($r[D]$, resp.)*

Notice that for disjunctions it is natural to define C, C' so that they condition the RGFs to be either existential (e.g., $s_{|z}$) or a collection of ground formulae. The necessity to condition prevents the generation of trivial disjunctive formulae. For conjunction and negation it is also possible to focus first on each element in x separately.

4.1.1 Nesting

The inductive definition 4.6 indicates that operands in the relational calculus may be RGFs themselves. This is a "symbolic" calculus that is always "compiled" into formulae in $\mathcal{R}$, however, and not a recursive operation. True recursion will be introduced in the next section.

4.1.2 Naming

Naming is a mechanism that allows the definition of new RGFs in terms of existing RGFs. The syntax of the operation is

$$A \doteq f(s_1, s_2, \ldots s_n),$$

where f is any operation in the relational calculus, including nesting. Given an instance x, A outputs identical formulae to those produced by $f(s_1, s_2, \ldots s_n)$. The use of mnemonics contributes only to simplify "programming" and analysis.

Example 4.5 *A relation that designates that a certain (or any) word is a modal, can be generated using an RGF that makes use of the word sensor and a conditioned version of the tag sensor. E.g., $word\&tag_{|tag(MOD)}$ would produce active formulae such as $word(can) \wedge tag(MOD)$ and $word(x) \wedge tag(MOD)$. Then, we may want to name it using a useful mnemonic: $MODAL \doteq word\&tag_{|tag(MOD)}$*

5 Structural Instance Space

So far we have presented $\mathcal{R}$ and RGFs with respect to an abstract domain D. In most domains more information than just a list of objects is available. We abstract this using the notion of a *structural domain* that we define below. Instances in a structural domain are augmented with some structural information and, as a result, it is possible to define more expressive RGFs in terms of the sensors provided along with the domain.

5.1 Structured Instances

Definition 5.1 *Let D be the set of elements in the domain. A* structured instance *O is a tuple $(V, E_1, E_2, \ldots E_k)$ where $V \subseteq D$ is a set of elements in the domain, and E_i is a set of edges on V. The graph $G_i = (V, E_i)$, is called the ith* structure *of the instance O and is restricted to be an acyclic graph on V.*

Example 5.1 (NLP) *Let D be the set of all words in English. A structured instance can correspond to a sentence, with V, the set of words in the sentence and $E = E_1$ describes the linear structure of the sentence. That is, $(v_i, v_j) \in E$ iff the word v_i occurs immediately before v_j in the sentence.*

Example 5.2 (VP) *Let D be the set of all positions in a 100×100 gray level image. A structured instance*

can correspond to a sub-image, such that V is the set of pixels in the sub-image and $E = E_1$ describes the top-down and left-right adjacencies in it. That is, $(v_j, v_k) \in E$ iff the pixel v_j is either immediately to the left or immediately above v_k.

The interaction of the system with the World is done via a collection of structured instances along with a set of sensors which can operate on them. Sensors may extract information directly from the input, such as word, tag, or intensity, but can also utilize outside knowledge in processing the input. More complex RGFs are then defined as functions of the sensors, and can be used to extract further information from the instances.

5.2 Structural Operations

We now augment the relational calculus of Section 4.1 by adding structural operations. These operations exploit the structural properties of the domain as expressed in the graphs G_is in order to define RGFs, and thereby generate non-atomic formulae that may have special meaning in the domain.

Definition 5.2 *Let $s_1, s_2, \ldots s_k$ be RGFs for $\mathcal{R}$. $colloc_g(s_1, s_2, \ldots s_k)$ is a restricted conjunctive operator that is evaluated on a chain of length k in the gth structure of the given structured instance. Specifically, let $O = \{G_i\}_1^m$ be a structured instance and let $v_1, v_2, \ldots v_k$ be a chain in G_i. The formulae generated by $colloc_i(s_1, s_2, \ldots s_k)$ are those generated by $s_1[v_1]\&s_2[v_2]\&\ldots\&s_k[v_k]$, where by $s_j[v_j]$ we mean here the RGF s_j focused to $\{v_j\} \subseteq D$, and the & operator is defined as in Definition 4.6. Notice that each RGF in the conjunctions may produce more then one formulae.*

Example 5.3 (NLP) *Say we are interested in a Subject-Verb relation, and assume that the structured instance consists of, in addition to the linear structure of the sentence (G_1), a graph G_2 encoding functional relationships among the words. Using a* role *sensor we can produce the Subject-Verb relation using the RGF $colloc_2(role_{|role(Subject)}, role_{|role(Verb)})$.*

Example 5.4 (VP) *To find an edge relation in an image, we might define an Edge RGF thus: $EDGE \doteq colloc_1(s, s, s, s, s)$ where s is, say, a sensor producing active relations for pixels with intensity value above* 50.

Definition 5.3 *Let $s_1, s_2, \ldots s_k$ be RGFs for $\mathcal{R}$. $scolloc_g(s_1, s_2, \ldots s_k)$ (*sparse colloc*) is a restricted conjunctive operator that is evaluated on a chain of length $n (k \leq n)$ in the gth structure of the given structured instance. Specifically, let $O = \{G_i\}_1^m$ be a structured instance and let $v_1, v_2, \ldots v_n$ be a chain in G_i. Formulae are generated by $scolloc(s_1, s_2, \ldots s_k)$ as follows: For each subset $v_{i_1}, v_{i_2}, \ldots v_{i_k}$ of elements in v, such that $i_j < i_l$ when $j < l$, all the formulas: $s_1[v_{i_1}]\&s_2[v_{i_2}]\&\ldots\&s_k[v_{i_k}]$, are generated, where by v_{i_i} we mean here the instance $\{v_{i_j}\} \subseteq D$, and the & operator is defined as in Definition 4.6.*

5.3 Focus-Word Centered Representation

The structural information also provides an easy way to *focus* the RGFs (Def. 4.5). For example, defining a set of elements for the focus set E in $s[E]$ can be done using some graph property. Specifically, we use the notion of a focus node, and define a focus set with respect to it using a radius length. In particular, in the *colloc*, *scolloc* operations, we can restrict the chains to *start* at a specific element $d \in D$ or to *contain* it. Notice that if, for the given instance $O = (V, G)$, we have that $d \notin V$, then the output is an empty set of formulae.

Example 5.5 *The power of the focus device can be shown in the following example for defining an RGF for* gender agreement *within an observation. Say we have a gender sensor, which will either output the relations* gender(male) *or* gender(female) *for gendered words or null otherwise. We can then define the RGF: $gender[focus]_{|gender[\sim focus]}$. That means that we are evaluating the gender RGF taking the focus element as the instance x, but that we are conditioning the output of this RGF to be active only when it produces a formula in the set of formulae produced by $gender[\sim focus]$. This essentially means that the RGF produces an active formulae of the form gender(male) or gender(female) iff the gender of the focus element matches the gender of another word in the sentence.*

5.4 Complexity of Evaluating RGFs

Notice that while all the formulae generated are in $\mathcal{R}$ and the structural operations only provides a way to specify which atomic formulas should be evaluated on which object, they actually provide a powerful mechanism that goes beyond $\mathcal{R}$. While variables in formulae in $\mathcal{R}$ have a single predicate in their scope, the structural properties provides a way to go beyond that (as shown in Ex. 5.5), but only in a restricted way that is efficiently evaluated.

This, of course, can be done within the language of first order logic by, say, adding predicates that encode the structural properties, e.g., an $edge(x, y)$ predicate.

This way, however, formulae will not be in $\mathcal{R}$ anymore, and the problem of evaluating a clause given an instance would become intractable. Structural operations defined on $\mathcal{R}$ allow us to define RGFs the constrain formulae evaluated on different objects without incurring the cost usually associated with enlarging the scope of free variables. This is done by enlarging the scope only as required by the structure of the domain, modeled by the G_is. Exploiting the structure allows for efficient evaluation of the formulae with the only additional cost being that of finding chain in the graph.

Proposition 5.1 *Let O be a structured instance, with $n = |V|$. Let t be an upper bound on the time to evaluate sensors in O.*

(i) The time to evaluate a conjunctive RGF of size k is bounded by $T_k = k \cdot n \cdot t$.

(ii) The time to evaluate a colloc RGF of size k is bounded by $c \cdot T_k$, where c is the number of chains of size k in the given graph.

(iii) The time to evaluate an scolloc RGF of size k is bounded by $c \cdot \binom{n}{k} \cdot T_k$, where c is the number of chains of size n in the graph.

Structural instances also provide the ability to deal with recursive RGFs. Although recursion is not supported in $\mathcal{R}$, it is inherited from the graph. Since a *path* relation in a graph is recursive over the *edge* relation, *colloc* and *scolloc* inherit recursion at no additional cost, providing, for example, the ability to generate "above" RGF from the"on" sensor [9].

6 An Example

The following example serves to illustrate the language and knowledge representations described in this paper. It utilizes basic sensors, structural operations with recursion, implicit and explicit focus definition, and conditioning. It is based on an implemented system that has been used to support learning for knowledge intensive tasks [19, 22, 17, 21].

The problem considered is learning *verb representations*. Once a system has learned a "definition" for when to use each verb it could decide, for example, to use *throw* rather than *remove* in a specific context [7, 18]. This problem is subtle, because the context for many verbs is similar enough that it may be hard to learn good representations for them only in terms of surrounding words and part of speech information. Verb representations are likely to depend on complicated relations that hold in the sentence and on semantic information [6].

6.1 Input

The input consists of a structured instance O that is given along with the sensors that can act on it. Let $O = \{V, G_1, G_2\}$ be the structured instance, with V - the set of all words in the sentence. G_1 corresponds to the linear structure of the sentence and G_2 to a structure encoding *functional dependencies* in the sentence [4]. The sensors given to operate on O consist of:

word - Outputs active relations of the form $word(a)$ where a is the spelling of a word $v \in V$. e.g. $word(dog)$, $word(locomotion)$, or $word(\mathbf{z})$.

suffix - Outputs active relations of the form $suffix(a)$ where a is the suffix of a word $v \in V$, or $suffix(\mathbf{z})$, when any one of a fixed set of suffixes is active.

role - Outputs active relations of the form $role(a)$ where a is the function of a word $v \in V$ in a functional dependency grammar. e.g. $role(Subject)$, $role(Main\_Verb)$, or $role(\mathbf{z})$.

6.2 Useful Relations

We now devise a set of "kinds" of relations that may be useful for learning verb representations. It is important to note that we only need to define "kinds" of relations, and not to explicitly define the relations themselves; it is clear that different instantiations of the "kinds" (RGFs) defined would be relevant for different verbs. Also, one can suggest a superset of "kinds", and allow the data driven learning process decide which are indeed relevant.

The following presentation is not complete, but would be enough to illustrate the approach. We denote by *target* the verb for which a representation is sought. Naturally only sentences in which *target* occurs are considered. Useful relations could include:

1. There exists a target word a.
2. There exists some suffix for the target word.

[4] This can be produced by a functional dependency grammar (FDG) of a sentence, which gives each word a specific function, and then structures the sentence hierarchically based on it. The graph is a tree that is rooted at the main verb of the sentence. This structure could be generated by an external rule-based parser or a learned one [17]; this issue is orthogonal to the current discussion which abstracts these issues into the sensors.

3. There exists a word a such that the subject of the target verb is a.

4. There exists a word a such that the direct object of the target verb is a.

Written explicitly, these relations are not in the language $\mathcal{R}$. However, they can all be defined as structural conjunctions of atomic formulae applied over a restricted domain. These conjunctions *will* be in $\mathcal{R}$, and allow the mapping to Boolean variables. We now present RGFs that produce relations of these forms.

6.3 Relation Generation Functions

Relations of the type (1) above, will be produced using the *word* sensor restricted to the *target* focus, written as $word[target]$.

The relations in (2) can be generated using the RGF

$$suffix_{|\mathbf{z}}[target].$$

This will create the existential relation $suffix(\mathbf{z})$ when the target word has been suffixed, but will not indicate what suffix it has. (It is possible to generate the ground version as well.)

For (3) we can use the RGF

$$colloc_2(role_{|role(Subject)}\&word, word[target]).$$

This RGF attempts to find a chain $v', v'' \in V$ in the second structure where the conjunction:

$$role_{|role(Subject)}[v']\&word[v']\&word[v''],$$

is active and $v'' = target$. If it does this, then it outputs an active formula of the form

$$role(Subject)\wedge word(a)\wedge word(b)$$

where a, b correspond to two words in V. Due to the colloc operation, a is the subject word, and b is the target word. Note that we can assume that the subject word will always precede the target word in G_2 due to the structure in the domain (i.e., the graph representing the dependency grammar).

The RGF for (4) is

$$colloc_2(role_{|role(Object)}\&word, word[target]),$$

which behaves as the one for (3), producing active formulae for all chains with objects pointing to the target.

Finally, let O correspond to the sentence:

The boy batted the ball,

with $target = batted$. The above RGFs would generate the following intermediate representation:

$\{word(batted),$

$suffix(\mathbf{z}),$

$role(Subject)\wedge word(boy)\wedge target,$

$role(Object)\wedge word(ball)\wedge target\}$

which are likely to be useful for learning verb representations. Note that these RGFs produce both quantified and ground formulae. The specific formulae produced depend on the data observed. As pointed out in Section 2, the consistency of the generation guarantees that similar relations that hold in different instances would be given the same identifiers and will thus facilitate generalization. In addition, the learning procedure would "decide" which of the formulae produced actually contribute to the sought after definition, and would produce a program that combines these appropriately.

7 Conclusion

This work describes a knowledge representation paradigm that is developed as part of a research program that develops a unified approach for knowledge representation, learning and inference. In particular, the representations studied here support efficient, data-driven generation of expressive representations that can be used by propositional learning algorithms. In addition to allowing the learnability of relational programs using propositional means, this approach provides a uniform way to abstract domain information that is available at a given stage of the learning process, using the notion of *sensors*. Domain information may be available from many sources and modalities and in many forms. These can all be viewed as sensors and integrated in a unified way into the developed paradigm.

Sensors also provide a clean solution from a software engineering point of view. A given learning architecture can be used to learn at several levels in the hierarchy, and for many domains. To do that, one has to supply it as input observations along with sensors that can interpret it. This approach has already been used successfully in several applications and is further developed and abstracted in [20]. This work opens up several directions for further work from the learning as well as the knowledge representation points of view; we hope that work along these lines would contribute to the development of a unified paradigm.

Acknowledgments

We are grateful to Jerry DeJong and Roni Khardon for useful comments on an earlier draft.

This research is supported by NSF grants IIS-9801638 and SBR-987345.

References

[1] A. Blum. Learning boolean functions in an infinite attribute space. *Machine Learning*, 9(4):373–386, October 1992.

[2] A. Carleson, C. Cumby, J. Rosen, and D. Roth. The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, UIUC Computer Science Department, May 1999.

[3] W. Cohen. PAC-learning recursive logic programs: Efficient algorithms. *Journal of Artificial Intelligence Research*, 2:501–539, 1995.

[4] W. Cohen. PAC-learning recursive logic programs: Negative result. *Journal of Artificial Intelligence Research*, 2:541–573, 1995.

[5] S. Dzeroski, S. Muggleton, and S. Russell. PAC-learnability of determinate logic programs. In *Proceedings of the Conference on Computational Learning Theory*, pages 128–135, Pittsburgh, PA, 1992. ACM Press.

[6] Y. Even-Zohar and D. Roth. Learning in NLP: Incorporating knowledge into the process. 2000. Submitted.

[7] A. R. Golding and D. Roth. A Winnow based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130, 1999. Special Issue on Machine Learning and Natural Language.

[8] D. Kapur and P. Narendran. NP-completeness of the set unification and matching problmes. In *Proc. of the 8th conference on Automated Ddeduction*, volume 230, pages 489–495. Springer Verlag, 1986.

[9] R. Khardon. Learning to take actions. *Machine Learning*, 35(1):57–90, 1999.

[10] R. Khardon and D. Roth. Learning to reason. *Journal of the ACM*, 44(5):697–725, Sept. 1997. Earlier version appeared in AAAI-94.

[11] R. Khardon and D. Roth. Learning to reason with a restricted view. *Machine Learning*, 35(2):95–117, 1999.

[12] R. Khardon, D. Roth, and L. G. Valiant. Relational learning for NLP using linear threshold elements. In *Proc. Int'l Joint Conference on Artificial Intelligence*, pages 911–917, 1999.

[13] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[14] J. W. Lloyd. *Foundations of Logic Progamming.* Springer-verlag, 1987.

[15] J. McCarthy. Programs with common sense. In M. Minsky, editor, *Semantic information processing*, pages 403–418. MIT Press, 1968.

[16] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 20:629–679, 1994.

[17] M. Munoz, V. Punyakanok, D. Roth, and D. Zimak. A learning approach to shallow parsing. In *EMNLP-VLC'99, the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, June 1999.

[18] J. Rosen. Scaling up context sensitive text correction. Master's thesis, UIUC, Department of Computer Science, May 1999.

[19] D. Roth. Learning to resolve natural language ambiguities: A unified approach. In *Proc. National Conference on Artificial Intelligence*, pages 806–813, 1998.

[20] D. Roth. Learning based programming. Technical Report UIUCDCS-R-99-2127, UIUC Computer Science Department, October 1999.

[21] D. Roth, M-H. Yang, and N. Ahuja. A SNoW-based face detector. In *NIPS-12*, 2000.

[22] D. Roth and D. Zelenko. Part of speech tagging using a network of linear separators. In *COLING-ACL 98, The 17th International Conference on Computational Linguistics*, pages 1136–1142, 1998.

[23] M. Sebag and C. Rouveirol. Any-time relational reasoning: Resource-bounded induction and deduction through stochastic matching. *Machine Learning.* To Appear.

[24] L. G. Valiant. Robust logic. In *Proceedings of the Annual ACM Symp. on the Theory of Computing*, 1999.

Experimental Results on Learning Soft Constraints

Alessandro Biso
Dipartimento di Informatica
Università di Pisa
Corso Italia 40
56125 Pisa, Italy

Francesca Rossi
Dipartimento di Matematica
Pura ed Applicata
Universita' di Padova
Via Belzoni 7
35131 Padova, Italy
frossi@math.unipd.it

Alessandro Sperduti
Dipartimento di Informatica
Università di Pisa
Corso Italia 40
56125 Pisa, Italy
perso@di.unipi.it

Abstract

Constraints are a very natural knowledge representation formalism. However, classical constraints (which are either satisfied or not) are not so flexible and cannot describe real-life features like preferences, costs, priorities, and uncertainties. Therefore recently many formalisms for soft constraints (which can be satisfied at a certain level) have been developed.

We address the problem of modeling a real-life problem by using soft constraints. In many real-life situations, one may know his/her preferences over some of the solutions, but have no idea on how to code this knowledge into the constraint problem in terms of local preferences, or also one may be able to give only a rough approximation of the desired levels of satisfaction of the constraints. We therefore suggest to treat the solution preferences as examples, and to employ a learning scheme which learns from such examples (either from scratch or from the available rough model) all the local preferences, so that the global preferences specified in the examples are satisfied. This paper presents experimental results on the use of this technique, presenting both a base version of the learning scheme and also an incremental anytime version, which helps in deciding when the amount of learning is enough.

1 Introduction

Classical constraint problems (CSPs) [16, 7, 8, 13, 14, 16, 15, 27] are a very expressive and natural formalism to specify many kinds of real-life problems. However, they also have evident limitations, mainly due to the fact that they are not very flexible when trying to represent real-life scenarios where the knowledge is not completely available nor crisp. In fact, in such situations, the ability of stating whether an instantiation of values to variables is allowed or not is not enough or sometimes not even possible. For this reason, many extensions of the classical CSP framework have been proposed in the literature. For example, fuzzy [5, 20, 23, 24], partial [9], probabilistic [6], hierarchical [4]. More recently, all these extensions have been unified in a general framework [2, 3], called SCSP, which uses a semiring (that is, a set plus two operations satisfying certain properties) to associate with each tuple of values for the variables an appropriate "degree of preference" (taken from the semiring). The two semiring operations define how to combine constraints together and how to compare different solutions.

Although being very expressive, soft constraint problems can make the modeling phase much more difficult, since one has to specify all preferences for every possible combination of values in each constraint. Moreover, sometimes one may know his/her preferences over some of the solutions but have no idea on how to code this knowledge into the constraint problem in terms of local preferences. That is, one has a global idea about the *goodness* of a solution, but does not know the contribution of each single constraint to such a measure. In such a situation, it is difficult both to associate a preference to the other solutions in a compatible way, and to understand the importance of each tuple and/or constraint. Another typical situation occurs when one has just a rough estimate of the local preferences, either for the tuples or for the constraints.

In [21] the first situation is theoretically addressed by proposing to use learning techniques based on gradient descent [22]. More precisely, it is assumed that the level of preference for some solutions (the *examples*)

is known, and a suitable learning technique is defined to learn, from these examples, values to be associated with each constraint tuple, in a way that is compatible with the examples.

In this paper we make the technique proposed in [21] concrete, we identify its features, and we show the results of several experiments run by choosing various values of these features. The SCSP problems on which the experiments are run are randomly generated, in a way which is similar to the random generation of classical CSPs [11].

Moreover, we also address the second scenario, by defining and experimenting with a generalized version of the basic learning technique which allows for incremental learning starting form a rough model of the constraint problem. The results obtained by this new technique show that indeed it is possible to use an incremental approach without loosing too much precision in the final result.

Notice that the learning problem we consider in this paper could be formulated also as a linear programming problem. However, the gradient descent technique allows us to perform the learning in an incremental and local way.

Another general framework for soft constraints, which is very related to SCSPs, is described in [25]. In practice what they do is to associate levels of preferences to constraints and not to tuples. The result is a syntactically different framework which however is equivalent to SCSPs when the order among the levels of preference is total [1]. Therefore all our results can apply to that framework as well in this special case.

The results are encouraging, and show that the proposed techniques (especially the incremental version) are both reasonable and promising. In fact, the final error in the levels of preferences which are learnt by our approach is small.

The learning approach that we use in this paper is based on gradient descent. The reason for this choice is that we believe that this is the simplest and most natural tool for our purposes. There could be other approaches to learning that may seem suitable for learning soft constraints, such as the one used in Inductive Logic Programming (ILP) [17]. However, although this framework has been recently extended to deal with probability distributions [18, 26], the ILP approach is not mature yet to deal with continuous domains, like semirings can be (for example when describing fuzzy constraints).

Overall, we can say that the problem we address in this paper, and the way we tackle it (by using the learning phase to model a constraint problem), can be cast within a general framework where reasoning and learning are not seen as separate tasks but are instead considered as strongly interacting with each other [12].

The paper is organized as follows. Section 2 gives the necessary notions about SCSPs. Then, Section 3 introduces the basic concepts of learning via gradient descent, and describes our learning approach. Section 4 describes in great detail all our experiments; in particular, Section 4.1 describes the the basic technique, and shows the corresponding results, Section 4.2 refers to the incremental technique, and Section 4.3 considers the case of non-representable functions. Finally, Section 5 concludes the paper and hints at possible directions for future work.

2 Soft constraint problems

Standard constraint satisfaction problems (CSPs) [15, 27] consist of a set of variables with a finite domain, plus a set of constraints. Each constraint involves a subset of the variables and specifies the tuples of values allowed for those variables. A solution for a CSP is then an assignment of values to all the variables such that all constraints are satisfied. Instead, in *soft* constraint problems a constraint does not say if a tuple is allowed or not, but rather at which level it is allowed. To describe such problems, in this paper we use the paradigm of semiring-based CSPs (SCSPs) [2, 3]: each tuple in each constraint is assigned a value (taken from the semiring), to be interpreted as the level of preference for that tuple, or its cost, or any other measurable feature. Then, constraints are combined according to the semiring operations, and the result of such a combination is that each assignment for all the variables has a corresponding semiring value too.

Semirings. A *semiring* S is a tuple $(A, +, \times, \mathbf{0}, \mathbf{1})$ such that

- A is a set and $\mathbf{0}, \mathbf{1} \in A$;
- $+$, called the additive operation, is a closed, commutative and associative operation such that $a + \mathbf{0} = a = \mathbf{0} + a$ (i.e., $\mathbf{0}$ is its unit element);
- $\times$, called the multiplicative operation, is a closed and associative operation such that $\mathbf{1}$ is its unit element and $a \times \mathbf{0} = \mathbf{0} = \mathbf{0} \times a$ (i.e., $\mathbf{0}$ is its absorbing element);
- $\times$ distributes over $+$ (i.e., $a \times (b + c) = (a \times b) + (a \times c)$).

Actually, we need to use a structure called *c-semiring*, which is just a semiring where + is idempotent, × is commutative, and **1** is the absorbing element of +.

Some properties. The idempotence of the + operation is needed in order to define a partial ordering $\leq_S$ over the set A, which enables us to compare different elements of the semiring. Such a partial order is defined as follows: $a \leq_S b$ iff $a + b = a$. Intuitively, $a \leq_S b$ means that a is "better" than b. This can be used to choose the "best" solution in our constraint problems. Both + and × are monotone on such ordering. The commutativity of the × operation is desirable when such operation is used to combine several constraints. In fact, were it not commutative, it would mean that different orders of the constraints give different results. If **1** is the absorbing element of the additive operation, then we have that $\mathbf{1} \leq_S a$ for all a. Thus **1** is the minimum (i.e., the best) element of the partial ordering. This implies that the × operation is *extensive*, that is, that $a \leq a \times b$. This is important since it means that combining more constraints leads to a worse (w.r.t. the $\leq_S$ ordering) result. The fact that **0** is the unit element of the additive operation implies that **0** is the maximum element of the ordering. Thus, for any $a \in A$, we have $\mathbf{1} \leq_S a \leq_S \mathbf{0}$.

Constraint systems and problems. A *constraint system* is a tuple $CS = \langle S, D, V \rangle$, where S is a c-semiring, D is a finite set, and V is an ordered set of variables. Given a constraint system where $S = (A, +, \times, \mathbf{0}, \mathbf{1})$, a *constraint* over CS is a pair $\langle def, con \rangle$, where $con \subseteq V$ and $def : D^k \to A$ (where k is the number of variables in con). A *constraint problem* P over CS is just a set of constraints over CS.

Combination and projection. The values specified for the tuples of each constraint are used to compute corresponding values for the tuples of values of all the variables, according to the semiring operations: × is used to combine the values of the tuples of each constraint to get the value of a tuple for all the variables, and + is used to choose the best among all such tuples. More precisely, we can define the operations of *combination* ($\otimes$) and *projection* ($\Downarrow$) over constraints. Consider two constraints $c_1 = \langle def_1, con_1 \rangle$ and $c_2 = \langle def_2, con_2 \rangle$ over CS. Then, their combination, $c_1 \otimes c_2$, is the constraint $c = \langle def, con \rangle$ with $con = con_1 \cup con_2$ and $def(t) = def_1(t \downarrow_{con_1}^{con}) \times def_2(t \downarrow_{con_2}^{con})$, where, for any tuple of values t for the variables in a set I, $t \downarrow_{I'}^{I}$ denotes the projection of t over the variables in the set I'. In words, the combination of two constraints c_1 and c_2 generates another constraint which involves all the variables of c_1 and c_2; each tuple of values for such variables, say t, gets a value of the semiring which is obtained by multiplying the values associated by c_1 and c_2 respectively to the two subtuples of t which are the projections of t over the variables of c_1 and c_2 respectively. Moreover, given a constraint $c = \langle def, con \rangle$ over CS, and a subset w of con, its projection over w, written $c \Downarrow_w$, is the constraint $\langle def', con' \rangle$ over CS with $con' = w$ and $def'(t') = \Sigma_{\{t | t \downarrow_w^{con} = t'\}} def(t)$. In words, the projection of a constraint c over a subset w of its variables is another constraint which involves only the variables in w; each tuple of values (for the variables in w), say t, gets a value of the semiring which is obtained by summing up all possible extensions of t to the other variables of c not in w.

Solutions. Given a constraint problem C over a constraint system CS, the *solution* of P is a constraint defined as $Sol(P) = (\bigotimes C)$: it is the constraint induced on all the variables by the whole problem. Such a constraint provides, for each tuple of values of D for all the variables, an associated value of A. More precisely, consider any tuple t with as many elements of D as the number of all variables (in the following such tuples will be called *n-tuples*). Then the corresponding value $val(t)$ can be obtained by (see definition of combination)

$$val(t) = def_1(t \downarrow_{con_1}^{V}) \times \ldots \times def_k(t \downarrow_{con_k}^{V}), \quad (1)$$

where V is the set of all variables, the set of all constraints is $C = \{c_1, \ldots, c_k\}$, and $c_i = \langle def_i, con_i \rangle$, for $i = 1, \ldots, k$. Given two n-tuples t and t', we say that t is better than t' if $val(t) \leq_S val(t')$. Note that this, by definition of $\leq_S$, means that $val(t) + val(t') = val(t)$. Note also that t and t' could be incomparable, since $\leq_S$ is a partial order. Given an SCSP P, we will call n-tuples(P) the set of all n-tuples of P. Then we will consider the function f : n-tuples(P) $\to A$ such that, for each n-tuple t, $f(t) = val(t)$. That is, the function which assigns to each n-tuple the corresponding value. We will call this f the *solution-rating function* of the given SCSP.

Example (concrete semirings). Let us recall the general form of an SCSP, which is $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$. As noted above, classical CSPs are just SCSPs where $A = \{true, false\}$, $+ = \vee$, $\times = \wedge$, $\mathbf{0} = false$, and $\mathbf{1} = true$. It is easy to check that the above described operations of constraint combination and projection reduce indeed in this case to the usual operations for the satisfiability of a set of constraints and for the elimination of some variables, respectively. Other interesting semirings can be thought. For example, the paradigm of fuzzy constraint problems (FCSPs)

[5, 20, 23, 24], where tuples get assigned values between 0 and 1 (to be interpreted as their level of preference), and the goal is to maximize the minimum level of preference, can be described by using the c-semiring where $A = \{x \mid x \text{ in } [0,1]\}$, $+ = max$, $\times = min$, $\mathbf{0} = 0$, and $\mathbf{1} = 1$. The last instance we cite is the one describing optimization problems, where each tuple has an associated cost and the goal is to minimize the global cost. Here the semiring is $\langle \Re^+, min, +, +\infty, 0 \rangle$. In the experiments reported in this paper we will use a subset of this semiring.

3 Learning soft constraint problems

Inductive learning [22] can be defined as the ability of a system to induce the correct structure of a map d which is known only for particular inputs. More formally, defining an example as a pair $(x, d(x))$, the computational task is as follows: *given a collection of examples of d, i.e., the training set, return a function h that approximates d.* Function h is called a hypothesis.

A common approach to inductive learning, especially in the context of neural networks, is to evaluate the quality of a hypothesis h (on the training set) through an *error function* [10]. An example of popular error function, that can be used over the reals, is the sum of squares error [10]: $E = \frac{1}{2}\sum_{i=1}^{n}(d(x_i) - h(x_i))^2$, where $(x_i, d(x_i))$ is the i-th example of the training set.

Given a starting hypothesis h_0, the goal of learning is to minimize the error function E by modifying h_0. This can be done by using a definition of h which depends on a set of internal parameters W, i.e., $h \equiv h_W$, and then adjusting these parameters. This adjustment can be formulated in different ways, depending on whether the domain is isomorphic to the reals or not. The usual way to be used over the reals, and if h_W is continuous and derivable, is to follow the negative of the *gradient* of E with respect to W. This technique is called *gradient descent* [10]. Specifically, the set of parameters W is initialized to small random values at time $\tau = 0$ and updated at time $\tau + 1$ according to the following equation: $W(\tau + 1) = W(\tau) + \Delta W(\tau)$, where $\Delta W(\tau) = -\eta \frac{\partial E}{\partial W(\tau)}$, and η is the step size used for the gradient descent.

Learning is stopped when a minimum of E is reached. Note that, in general, there is no guarantee that the found minimum is global.

Learning in our context can be used to find a suitable set of values to be assigned to the constraint tuples of an SCSP. More precisely, consider a CSP P. We recall that a CSP can be seen as an SCSP where the semiring is $\langle \{true, false\}, \vee, \wedge, false, true \rangle$. That is, allowed tuples are assigned the value *true*, and forbidden ones the value *false*. Given such a CSP, its solutions are those n-tuples t such that $val_P(t) = true$.

We assume now that for some solutions of P we are given a desired rating $d(t)$ defining the goodness of t. Let m be the number of such solutions. That is, we have the set of *examples* $TR = \{(t_1, d(t_1)), \ldots, (t_m, d(t_m))\}$. However, giving the examples is not enough, because we must also have an idea of how to combine and compare the values given as examples. Therefore, together with the examples, we must also be given the following two objects: a semiring containing the values in the examples, and a distance function over such a semiring. For our experiments we have chosen the semiring $\langle \Re^+, min, +, +\infty, 0 \rangle$ and, as distance function, the usual one over reals: $dist(val_{P'}(t), d(t)) = | val_{P'}(t) - d(t) |$. Once we have the above, our goal is to define a corresponding SCSP P' over the same semiring such that P and P' have the same graph topology, and for each n-tuple t such that $(t, d(t))$ is an example, $dist(val_{P'}(t), d(t)) < \epsilon$, where $\epsilon > 0$ and small.

Given the first condition (on the graph topology), the only free parameters that can be arbitrarily chosen in order to satisfy the other conditions are the values to be associated to each constraint tuple. For each constraint $c_i = \langle def_i, con_i \rangle$ in P, consider $S_i = \{t_{ij}$ such that $def_i(t_{ij}) = true\}$ (that is, the set of tuples allowed by that constraint). The idea is to associate, in P', a free parameter w_{ij} (note that w_{ij} must belong to the set of the chosen semiring) to each t_{ij} in S_i. With the other tuples, we associate the constant $\mathbf{0}$, which, we recall, is the worst element of the semiring (w.r.t. $\leq_S$). Given this association, the value assigned to each n-tuple t in P' is

$$val_{P'}(t) = \prod_{i=1}^{k}(\sum_{j=1}^{|S_i|} \text{subtuple}(t_{ij}, t, i) \times w_{ij}), \qquad (2)$$

where $\prod$ refers to $\times$, k is the number of constraints in P, $\sum$ refers to $+$, and $\text{subtuple}(t_{ij}, t, i) = \mathbf{1}$ if $t \downarrow^V_{con_i} = t_{ij}$ and $\mathbf{0}$ otherwise.

Note that, for each i, there is exactly one j such that $\text{subtuple}(t_{ij}, t, i) = \mathbf{1}$. Let l_i be such a j. Thus $\sum_{j=1}^{|S_i|} \text{subtuple}(t_{ij}, t, i) \times w_{ij} = w_{il_i}$, and therefore $val_{P'}(t) = w_{1l_1} \times \ldots \times w_{kl_k}$, which corresponds to equation 1. The values of the free parameters w_{ij} may be obtained via a minimization of the error function, which will be defined according to the distance function of the semiring. When minimizing such a function, one has to assure that, every time the values

of the free parameters are changed in order to make the error function smaller, the new values are in the semiring set as well. In other words, no change can be made to a value which is already **1** (resp., **0**) and which the learning technique would like to make even smaller (resp., bigger).

Assume that the desired ratings are real numbers and that the chosen c-semiring is $\langle \Re^+, min, +, +\infty, 0 \rangle$. Here $\times$, that is, standard sum, is continuous and derivable. Thus we can use the gradient descent technique.

In this case equation (2) becomes $val_{P'}(t) = \sum_{i=1}^{k}(min_{j=1,\ldots,|s_i|} \text{ subtuple}(t_{ij}, t, i) + w_{ij})$ and the error function is $E = \frac{1}{2}\sum_{i=1}^{m}(d(t_i) - val_{P'}(t_i))^2$. By deriving the error function w.r.t. one free parameter w_{ij} we get

$$\Delta w_{ij}(\tau) = -\eta \frac{\partial E}{\partial w_{ij}(\tau)} =$$

$$= -\eta \cdot \sum_{i=1}^{m}((d(t_i) - val_{P'}(t_i)) \cdot output(u_{ij})).$$

Thus only those weights which are associated to tuples which are contained in at least one n-tuple in an example will be changed. We recall that one has to make sure that all weights remain positive. This can be done by modifying the learning rule in such a way that weights that would become negative are not modified. See [19] for an example of this technique.

Notice that, for this semiring for constraint optimization problems, the learning problem could also be formulated as a quadratic programming problem. However, the gradient descent technique is in our view preferable, mainly for its locality and for the possibility of developing an incremental learning technique (see following sections). Also, in this case the gradient descent does not have any local minima, so we can be sure that stabilization is achieved only on globally optimal points.

4 Experimental results

As anticipated in the previous section, for now we have focussed our experiments on the semiring of constraint optimization over the reals, that is, $\langle \Re^+, min, +, +\infty, 0 \rangle$. The reason is twofold: first, on this semiring the distance function is already defined and the two operations are (or can be modified to be) continuous and derivable; second, constraints over this semiring can be used to build constraint optimization problems, and thus are interesting and significative enough.

To evaluate the behavior of the proposed learning technique, we have considered different learning scenarios. In the first case, we have tried to establish if the technique is able to reconstruct a solution-rating function starting from examples for a soft constraint problem of which we know the parameters. This is the simplest case, since we are sure that the function is representable by the model. To consider different conditions in a structured way, we used a variant of the notion of randomly-generated constraint problems. More difficult cases were analyzed as well. One deals with non-representable functions and the other one simulates a situation in which the user provides imprecise ratings. Both these scenarios are implemented by adding Gaussian noise to the desired ratings. For all these scenarios, we have considered two variants: one in which the system is given all the examples at once, and another, more realistic, one, in which examples are given on demand. Notice that the second variant of the learning technique can be used also to refine an approximate model of the constraint problem, obtained for example by giving preferences to constraints and not to tuples, or by assigning approximate preferences to the tuples.

The randomly-generated SCSP problems we used for our experiments are generated in a way which is very similar to the usual randomly generated CSPs [11], which are characterized by four parameters: $\langle n, m, p_1, p_2 \rangle$, where n is the number of variables, m is the size of the variable domains, p_1 is the density of the problem, that is, the percentage of constraints over the maximum number of constraints, and p_2 is the tightness, that is, the percentage of forbidden pairs in each constraint. To adapt this model to randomly generated SCSPs, we just need to add the semiring to be used and to decide what we mean by "forbidden pairs". In fact, while in standard CSPs pairs are either forbidden or allowed, in SCSPs each pair is allowed, but with a certain level of preference. Then we can decide either that "forbidden" means that its level of preference is not the best semiring value, or that it is the worst value. We have chosen the second interpretation, and thus in our framework p_2 is the percentage of pairs of each constraint which have the worst semiring value[1]. Therefore, once the semiring is fixed, a four-tuple $\langle n, m, p_1, p_2 \rangle$ is used to generate an SCSP with n variables, m values for each variable, $p_1\%$ of constraints, and for each constraint $p_2\%$ of pairs which have the worst semiring value. All the other pairs are given a semiring value randomly.

Our experiments have been performed on the semiring of constraint optimization over the reals, that is,

[1] We also have conducted some experiments with the other notion of tightness, obtaining similar results.

$\langle \Re^+, min, +, +\infty, 0 \rangle$. For practical reasons, it is usually preferable for the gradient descent technique not to have to deal with $+\infty$. Therefore, one can normalize the semiring $\langle \Re^+, min, +, +\infty, 0 \rangle$ by projecting $\Re^+$ over the interval $[0, x]$, with $x \neq +\infty$ representing $+\infty$, and adjusting the two operations accordingly. In our experiments, we have arbitrarily chosen $x = 11$. Thus p_2 is the percentage of pairs of each constraint which get the semiring value 11.

4.1 Basic version for representable functions

In the experiments reported in this paper, we first focus on the case in which the unknown function can be computed by the model, and we evaluate how well the proposed gradient descent technique can approximate such a function. To model this situation, we randomly generate an SCSP, which will be used as our model: the values of its n-tuples represent the unknown function, to be found by the learning technique. Therefore, some of its solution ratings will constitute the *training set* (that is, the examples), while others will be used as the *test set*, which is the set against which the results of the learning phase will be tested. As usual practice in learning, training set and test set are disjoint.

The n-tuples belonging to the training set are not chosen randomly, but in such a way that each pair in each constraint appears in at least one of the n-tuples taken as examples. This means that, for each parameter, we have at least some data that can help us to learn its value. However, we take more examples than these, but always in the order of the number of parameters of the problem, which is $\frac{p_1}{100} \times \frac{n \times (n-1)}{2} \times m^2$, while the size of the domain of the unknown function is n^m (that is, the number of all n-tuples of the problem).

Once chosen the training and the test set, we take the SCSP, we forget the semiring values associated to the pairs in the constraints, we assign to each pair a parameter, and we randomly initialize these parameters (to a value between 0 and 3). Then we start the learning phase, which, at each iteration, checks the value of the error function on the training set, adjusts the step size accordingly (following an adaptive gradient descent scheme), and changes the values of the parameters, until the maximum error on the training set goes below a certain threshold. The threshold we chose for our experiments is 1% of the maximum semiring value associated to an example in the training set. The step size is used as follows: at each iteration, to each parameter we sum the gradient multiplied by the step size; then, if the total error decreases, we augment the step size (by multiplying it by 1.1), otherwise we make its smaller (by multiplying it by 0.5). At the beginning, the step size is set to 0.5. The intuition underlying this *adaptive* technique for the step size is that a smaller step size allows for a better gradient descent. Therefore, when we are making a mistake (that is, the error grows), it is necessary to choose the correct direction for the descent, and thus it is appropriate to have a relatively small step size.

	Mean error			# examples
	t20%	t50%	t80%	
d20%	1.71	1.48	0.60	781
d40%	1.55	1.34	0.60	1545
d60%	1.25	1.17	0.55	2306
d80%	1.32	1.09	0.59	3068

Table 1: Mean error and number of examples.

The results we show are related to several SCSPs, obtained by considering four values of density (20%, 40%, 60%, and 80%) and three values of tightness (20%, 50%, and 80%). The number of variables is always 20 and they have 5 values each. Moreover, for each choice of a density and a tightness, we have generated three SCSPs, and we have computed the arithmetic mean of the results over such three SCSPs.

Table 1 shows the mean error on the test set for each density/tightness combination, and also the number of examples for each density. Notice that this is the relative and not the absolute error, since we relate the error to the range of all the correct values on the training set. Therefore, saying that a solution in the test set has an error equal to 1% means that the absolute error is $x/100$, where x is the size of the actual range of correct values. The most important thing to notice here is that in general the error is reasonable. Another point is that, given a certain density, the error decreases as the tightness increases. Moreover, given a certain tightness, the error slightly decreases with the density.

4.2 Incremental version for representable functions

Even though the results reported in Table 1 seem reasonable and thus show the feasibility of the proposed technique, one may be worried about the amount of examples needed to achieve such results. In fact, although the order of magnitude of the training set is always much smaller than the domain of the unknown function, still it may be too much for the user to give all such examples at once. For this reason, and also to be able to refine an approximate model, we have developed a variant of the basic technique proposed in

the previous section, where the user gives a bunch of examples at a time, on request.

The basic idea is the following. We want to model a realistic situation in which the system uses only a small quantity of examples, and then the user check the results and gives new examples, which represent situations which have not been learnt well by the system. And so on until all situations of interest to the user have been learnt satisfactorily.

The way we model this interactive process consists of performing several phases of learning, where each learning phase uses the same algorithm as in Section 4.1, but has different training and test sets. If the entire training set is called T, we consider its partition into a small part S of n elements, and the rest: $R = T - S$. Moreover, we also consider a subset I of R, again with n elements. The first learning phase will take S as its training set, and I as its test set. At this point, the elements of I which are not learnt well, say $bad(I)$, are then inserted into S to produce a larger training set for the next learning phase. Thus the new training set is $S \cup bad(I)$ and the new test set consists of n elements of $T - (S \cup I)$. Notice that the examples inserted into S will always be used in the later learning phases. This process is repeated until all elements of the current set I have been learnt well. At this point, the current size of the training set tells us how many examples we need in total from the user. One final issue of this technique is how to define the set $bad(I)$: what we do is to use a threshold (1% of the absolute error), which tells us which elements have not been learnt well (those for which the error is above the threshold). Obviously, the final result of the learning phase is then evaluated on a disjoint set of examples, which is the *real* test set for the overall experiment.

In our experiments with this incremental technique, we have chosen just one density (40%) and one tightness (50%), and the usual size for the variable set (20) and the domain size (5). Then we have generated the usual training set, and we have treated it as described above. However, the above description leaves some notions under-defined, and thus we run our experiments with all the possible choices, to check whether these choices influence the results or not. In particular:

- the value of n, that is, the initial size of the training set and the size of the set I at each time, has been set to 150 in one set of experiments, and to 300 in another set, to check the influence of this parameter on the resulting error.
- Another source of possible nondeterminism could be the way in which the training set is "visited" during the whole process, that is, which elements are considered first and which later. In fact, it could be that the elements considered at the beginning produce an overfitting, that is, a specialization over such elements which could interfere with the correct learning of the elements considered later. Therefore we run our experiments both reading the training set from the beginning to the end, and also from the end to the beginning.
- The incremental learning technique described above does not say if, after each enlarging of the training set, the next learning phase must start from scratch or from where the previous learning phase has ended. One might think that the final results could be very different, since starting from scratch could mean forgetting what we have learnt before and thus avoiding a possible overfitting over the examples of the training set of the previous phase. Our experiments have shown that the two alternatives produce almost equivalent results, but starting from scratch is less efficient. Thus we will not show the results for this case. Notice that this observation shows the feasibility of the application of our incremental technique when starting from an approximate model.

Summarizing, we have two different choices when running an experiment: $n = 150$ or $n = 300$, and forward or backward visit of the training set. We show the results for the forward visit for both values of n, while the backward visit is shown only for $n = 150$ (results for $n = 300$ are very similar).

Figures 1 and 2 show the number of examples in the training set at each phase and the mean error in percentage, respectively. The first thing to notice is that

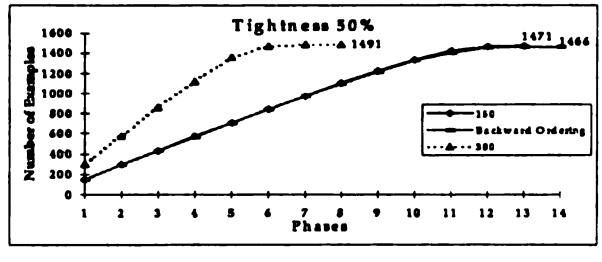


Figure 1: Number of examples for each phase.

the way the training set is visited is not relevant, since both the number of examples and the mean error stay the same. Also, when $n = 300$, the training set grows faster in size than when $n = 150$. However, at the end of the learning procedure, the training sets have roughly the same size. On the contrary, when considering the mean error, for $n = 300$ the error decreases

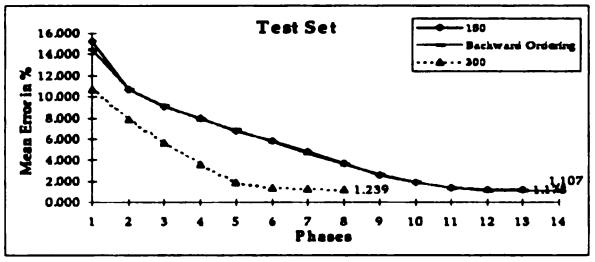


Figure 2: Mean error in each phase.

faster. However, even for this measure, the final values are similar. From this observations, it may seem that the incremental learning procedure is independent from the value of n. However, if one considers the computational cost (see Figure 3), this measure is greater for $n = 150$, since one has to perform a greater number of phases to obtain the same result as for $n = 300$.

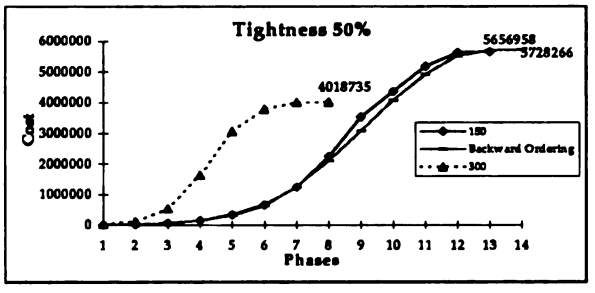


Figure 3: Cumulative computational cost.

4.3 Non-representable functions

We have also simulated the situation in which the desired solution-rating function is not representable in the underlying model. This may happen in our framework, since we consider given a certain constraint graph, which may not be able (whatever the constraint definitions are) to produce the desired solution rating function. For example, the user may give us contradictory ratings for correlated solutions, which means that the constraint graph is not adequate for the user's expectations.

We have modeled this situation by adding noise to the training set and also to the test set. This noise follows a Gaussian model with mean 0 and deviation which is 5% of the highest desired value. Even for this scenario, as we did for representable functions, we experimented with both our basic learning technique and the incremental one. Because of the presence of noise, here it is not possible to use the same stop criterion as before (error smaller than a threshold), thus we continue

	Mean error					
	Test with noise			Test without noise		
	t20%	t50%	t80%	t20%	t50%	t80%
d20%	4.80	4.84	5.38	4.21	4.05	3.78
d40%	4.79	5.88	5.82	4.07	4.69	3.94
d60%	5.49	6.21	6.95	4.49	4.81	4.46
d80%	6.26	6.54	7.41	5.17	5.06	4.59

Table 2: Mean error for non-representable functions.

the learning process until no significant improvement is shown.

The first three columns of Table 2 show the mean error in percentage on the test set for the basic learning procedure. It is easy to see that the error values are much higher than in the representable case. However, the learning process shows a rather robust behavior, since, apart from the higher values, it behaves similarly to the representable case.

Figures 4 and 5 show, for the incremental technique, the number of examples needed for the phases and the mean error in such phases, respectively. These figures do not show what happens when visiting the training set backwards, since we have seen (by running experiments) that the direction of the visit does not influence the result. Again, we may note that these curves are similar in shape to those for the representable case, but show higher error values, as expected.

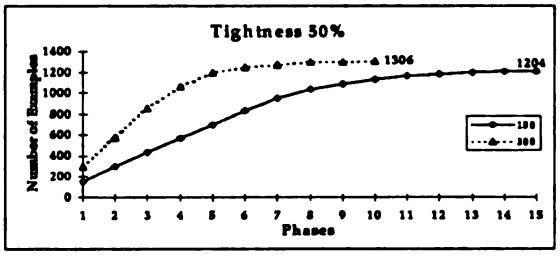


Figure 4: Incremental version: number of examples for each phase.

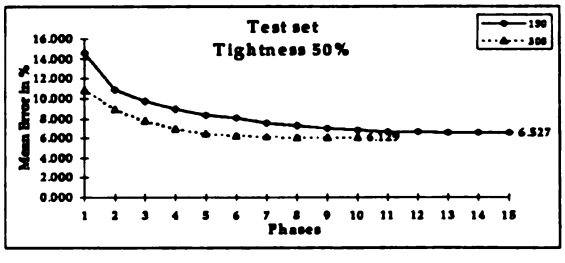


Figure 5: Incremental version: mean error in each phase.

Further experiments were run to simulate the case of a user which gives imprecise knowledge about the desired solution ratings. This scenario was modeled by the introduction of noise only on the elements of the training set (and not on the test set). The last three columns of Table 2 show the mean error on the test set. Notice that it is lower than for non-representable functions (due to the absence of noise on the test set), but otherwise has a similar relationship with density and tightness. The curve of the mean error in the various phases (not shown) is similar in shape to that in Figure 5 but leads to smaller values: for $n = 150$ the minimum error is 5.514 (instead of 6.527) and for $n = 300$ it is 4.991 (instead of 6.129).

As for the computational cost, Figure 6 shows results that are similar to those in Figure 3, except that there is no saturation at the end of the curves. This may be due to the presence of noise, or also to the use of an error threshold which is too strict.

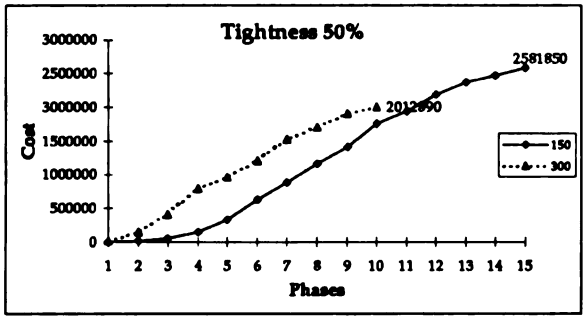


Figure 6: Cumulative computational cost for non-representable functions.

Summarizing, the system seems to have responded well to the introduction of noise, especially in the case which simulates an imprecise user. In fact, the noise introduced was rather relevant (up to 5%), and the error values are very close to this value. This means that the system was able to substantially learn the desired function (except for the noise).

5 Conclusions and future work

We have run several experiments which simulate the task of learning an SCSP, given a fixed constraint graph and some examples of solution ratings. These experiments have been run on a set of randomly generated SCSPs, and have been evaluated by studying the mean error on a test set for the solution ratings.

We also developed a variation of the basic learning technique which allows to get a reasonable result without the need to give all the examples at once, thus making the user role in the modeling phase even easier. This new technique is based on an incremental and interactive process, composed of several learning phases in sequence. Finally, we have considered functions which are not representable in the model, simulating this lack of precision by introducing noise in the learning technique.

The results obtained so far are encouraging and show that the proposed technique is promising. We plan to study its behavior on larger random problems and real (that is, not randomly generated) problems, and to extend the learning capability to discrete semirings.

References

[1] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based CSPs and valued CSPs: Basic properties and comparison. In *Over-Constrained Systems*. Springer-Verlag, LNCS 1106, 1996.

[2] S. Bistarelli, U. Montanari, and F. Rossi. Constraint Solving over Semirings. In *Proc. IJCAI95*. Morgan Kaufman, 1995.

[3] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Solving and Optimization. *Journal of the ACM*, 44(2):201–236, March 1997.

[4] A. Borning, M. Maher, A. Martindale, and M. Wilson. Constraint hierarchies and logic programming. In Martelli M. Levi G., editor, *Proc. 6th International Conference on Logic Programming*, pages 149–164. MIT Press, 1989.

[5] D. Dubois, H. Fargier, and H. Prade. The calculus of fuzzy restrictions as a basis for flexible constraint satisfaction. In *Proc. IEEE International Conference on Fuzzy Systems*, pages 1131–1136. IEEE, 1993.

[6] H. Fargier and J. Lang. Uncertainty in constraint satisfaction problems: a probabilistic approach. In *Proc. European Conference on Symbolic and Qualitative Approaches to Reasoning and Uncertainty (ECSQARU)*, pages 97–104. Springer-Verlag, LNCS 747, 1993.

[7] E. C. Freuder. Synthesizing constraint expressions. *Communication of the ACM*, 21(11), 1978.

[8] E. C. Freuder. Backtrack-free and backtrack-bounded search. In Kanal and Kumar, editors, *Search in Artificial Intelligence*, pages 343–369. Springer-Verlag, 1988.

[9] E. C. Freuder and R. J. Wallace. Partial constraint satisfaction. *AI Journal*, 58, 1992.

[10] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991.

[11] T. Hogg, B.A. Hubermann, and C.P. Williams, editors. *Special Volume on Frontiers in Problems Solving: Phase Transition and Complexity.* Artificial Intelligence, vol.81, n. 1-2, 1996.

[12] R. Khardon and D. Roth. Learning to reason. *Journal of the ACM*, 44(5):697–725, September 1997.

[13] A. K. Mackworth and E. C. Freuder. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25:65–74, 1984.

[14] A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[15] A.K. Mackworth. Constraint satisfaction. In Stuart C. Shapiro, editor, *Encyclopedia of AI (second edition)*, volume 1, pages 285–293. John Wiley & Sons, 1992.

[16] U. Montanari. Networks of constraints: Fundamental properties and application to picture processing. *Information Science*, 7:95–132, 1974. Also Technical Report, Carnegie Mellon University, 1971.

[17] S. Muggleton. Inductive logic programming: issues, results and the lll challenge. *Artificial Intelligence*, 114(1-2):283–296, December 1999.

[18] S. Muggleton. Stochastic logic programs. *Journal of Logic Programming*, To appear.

[19] M. Pelillo and M. Refice. Learning compatibility coefficients for relaxation labeling processes. *IEEE Transactions on Patterns Analysis and Machine Intelligence*, 1994. vol.16, n.9.

[20] A. Rosenfeld, R.A. Hummel, and S.W. Zucker. Scene labelling by relaxation operations. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(6):420–433, 1976.

[21] F. Rossi and A. Sperduti. Learning solution preferences in constraint problems. *Journal of Experimental and Theoretical Computer Science*, 1998. Vol 10.

[22] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

[23] Zs. Ruttkay. Fuzzy constraint satisfaction. In *Proc. 3rd IEEE International Conference on Fuzzy Systems*, pages 1263–1268, 1994.

[24] T. Schiex. Possibilistic constraint satisfaction problems, or "how to handle soft constraints?". In *Proc. 8th Conf. of Uncertainty in AI*, pages 269–275, 1992.

[25] T. Schiex, H. Fargier, and G. Verfaille. Valued Constraint Satisfaction Problems: Hard and Easy Problems. In *Proc. IJCAI95*, pages 631–637. Morgan Kaufmann, 1995.

[26] M. Sebag and C. Rouveirol. Tractable induction and classification in first order logic via stochastic matching. In *Proc. IJCAI97*, pages 888–893. Morgan Kaufman, 1997.

[27] Edward P. K. Tsang. *Foundations of Constraint Satisfaction.* Academic Press, 1993.

Propositional Logic and One-Stage Decision Making

Hélène Fargier **Jérôme Lang**
IRIT
Université Paul Sabatier
31062 Toulouse Cedex (France)
{fargier, lang}@irit.fr

Pierre Marquis
CRIL
Université d'Artois
62307 Lens Cedex (France)
marquis@cril.univ-artois.fr

Abstract

In this paper, a pure propositional framework for representing qualitative, one-stage decision problems under uncertainty is presented. The representation is based on a partition of the propositional variables between state variables – observable or not – and decision variables, and a distinction between pieces of knowledge, constraints, and goals. Effects of decisions are computed using dependency-based update and the Modified Possible Models Approach (MPMA). Various assumptions about the structure of the problem (especially observability), and the way of computing the consequence of a decision, are considered. We are specifically interested in finding out decision policies, i.e., determining for each possible observation a decision that ensures to reach a goal state if taken (whatever the unobservable variables). The existence of a solution policy is investigated from the point of view of computational complexity. The search for a policy is shown to be related to the resolution of the validity problem for some quantified boolean formulas (QBF), and to the existence of some prime implicants modulo a formula.

1 Introduction

Informally, decision making under uncertainty consists in finding suitable actions (possibly in sequence, possibly conditionally on observations) so as to be likely enough to reach a satisfactory state.

In classical decision theory (e.g., [33]) the data of a given problem are generally represented in an extensive, space-consuming form by listing all states, all actions, all possible consequences, etc. This is where AI (and more specifically logic) has a role to play, because it has contributed to the development of representation languages which can encode a decision problem in a much more structured and compact way. Actually, logic can be used in (at least) three different ways. The first one is the description of the system (its initial state, the available decisions and their effects) using expressive knowledge representation languages (especially for reasoning about actions). The second one is the description of the preferences of the agent, through preference representation languages. The third one is the actual computation of decision policies using automated reasoning techniques (such as theorem proving, model finding, abduction).

In this paper, we make the following assumptions about the decision process and its representation:

1. *structured decisions.* Not only the *state* space is represented in a compact, structured way, but so is the *decision* (or *action*) space. Unlike in Markov Decision Processes and most planning formalisms, decisions consist of simultaneous truth assignments of a number of decision variables (consider for instance the cases where assigning a decision variable to *true* or *false* consists in pushing a button or not, or, taking or not a medicine). To express that some particular assignments are (physically) impossible, the data include a set of *constraints* bearing on decision variables.
2. *partial observability* of the initial state of the system. Specific cases of it are full observability, and, at the opposite side, unobservability.
3. *nondeterministic effects of decisions.* Effects of decisions are encoded using the recent framework for belief update named *Modified Possible Models Approach* (MPMA) [10], which offers a much more satisficing handling of disjunctive updates than Winslett's standard PMA [35]. An additional strong point of the MPMA compared with

the PMA is its complexity (which is an extra-result offered by this paper): coNP-complete versus Π_2^p-complete [12].

4. *purely qualitative expression of knowledge and goals.* Everything is encoded in pure propositional logic. Especially, binary preferences are considered, only: states are partitioned among goal states and non-goal states and the purpose of the agent is to find out decisions enabling to reach a goal state when taken. No graded uncertainty about the possible outcomes of an action (e.g., a probability distribution over the resulting states) is taken into account.

5. *one decision stage.* This simplification not only makes the exposition simpler, but it is also practically relevant because (i) many problems involve only one decision stage and (ii) our framework can be easily extended (see in Conclusion) to multi-stage decision *without observability.*

Points (1) (2) and (3) above offer enough flexibility to make our framework practically relevant. Especially, propositional STRIPS and several generalizations of it can be viewed as specific instances of this framework. Points (4) and (5) can be considered quite restrictive; nevertheless, (4) enables a pure propositional logic framework for decision making; hence it is possible to take advantage of various well-known computational techniques of this framework, like prime implicants algorithms. Another motivation for (4) and (5) is the necessity to focus on some specific cases of general decision making so as to identify precisely its sources of complexity (the existence of a plan in the propositional STRIPS framework is already PSPACE-complete [5]).

The problem of qualitative decision making considered in this paper is the following: *is there a policy assigning a decision to each possible observation that ensures that a goal state will be reached?* The purpose of this paper is to study both the existence of such a policy (especially from a computational complexity point of view) and the practical computation of policies fulfilling the above condition *as much as possible.* The search for a solution policy for a qualitative decision problem is shown related to the validity problem for *quantified boolean formulas*, or QBF (studied mainly in the area of complexity theory [34] [29]). It is also related to the existence of some specific prime implicants modulo a formula. Based on this connection, an algorithm for computing policies is presented.

2 Background and notations

2.1 Propositional logic

Let PS be a finite set of propositional variables. $PROP_{PS}$ is the propositional language built up from PS and the connectives in the usual way. For every $X \subseteq PS$, $PROP_X$ denotes the sublanguage of $PROP_{PS}$ generated from the variables of X, only. $\top$ denotes the boolean constant that is always true, and $\bot$ the boolean constant that is always false. For every formula Σ of $PROP_{PS}$, $Var(\Sigma)$ is the set of variables occurring in Σ. $\Sigma[x \leftarrow \top]$ (resp. $\Sigma[x \leftarrow \bot]$) denotes the formula from $PROP_{PS}$ obtained by substituting in a uniform way the variable $x \in PS$ by the boolean constant $\top$ (resp. $\bot$) in Σ. Full instantiations of variables of $X \subseteq PS$ (resp. PS) (worlds) are denoted by $\vec{x}$ (resp. $\vec{\omega}$). Sometimes we identify the complete assignment $\vec{x}$ with the usual formula of $PROP_X$ formed by taking the conjunction of all X-literals as assigned by $\vec{x}$. 2^X denotes the set of all possible assignments of variables of X. If X and Y are two disjoint subsets of $PROP_{PS}$ then $(\vec{x}, \vec{y})$ is the assignment of $2^{X \cup Y}$ that assigns any variable of X like $\vec{x}$ does and any variable of Y as $\vec{y}$ does. We shall also identify finite sets of formulas with the conjunctions of their elements each time this will be convenient.

The set $PI_\Sigma^X(\Phi)$ of prime implicants modulo a formula Σ of a formula Φ over $PROP_X$ is defined (up to logical equivalence) by $PI_\Sigma^X(\Phi) = PI^X(\Sigma \Rightarrow \Phi) \setminus PI^X(\neg\Sigma)$, where $PI^X(\Phi) = PI(\Phi) \cap PROP_X$ and $PI(\Phi)$ $(= PI_\top^{PS}(\Phi))$ denotes the set of prime implicants of Φ. For instance, if $\Sigma = \{a \vee b, (a \wedge c) \Rightarrow e, d \Rightarrow e\}$, then $PI_\top^{\{a,b,c,d,e\}}(\Sigma) = \{a \wedge e, a \wedge \neg c \wedge \neg d, b \wedge e, b \wedge \neg c \wedge \neg d, \neg a \wedge b \wedge \neg d\}$; $PI_\top^{\{a,b,c,d,e\}}(\neg\Sigma) = \{\neg a \wedge \neg b, a \wedge c \wedge \neg e, d \wedge \neg e, \neg b \wedge c \wedge \neg e\}$; $PI_\Sigma^{\{a,b,c,d\}}(e) = \{a \wedge c, d, \neg b \wedge c\}$; $PI_\Sigma^{\{a,c\}}(e) = \{a \wedge c\}$; $PI_\Sigma^{\{a,b\}}(e) = \emptyset$; $PI_\Sigma^{\{a,b,c,d\}}(a \vee b) = \{\top\}$.

2.2 Computational complexity and QBF

In this paper we refer to some complexity classes above NP and coNP, details about which can be found in [29]. We assume that the classes P, NP and coNP are known to the reader. $\Sigma_2^p = \mathsf{NP}^{\mathsf{NP}}$ is the class of all languages recognizable in polynomial time by a nondeterministic Turing machine equipped with an NP oracle telling in unit time whether a given propositional formula is satisfiable or not; $\Pi_2^p = \mathrm{co}\Sigma_2^p$; $\Pi_3^p = \mathsf{coNP}^{\Sigma_2^p}$. The canonical Σ_2^p-complete problem is 2-QBF (where QBF stands for "quantified boolean formula"). An instance of 2-QBF consists of a triple $\langle A, B, \Phi \rangle$ where A and B form a partition of the set of variables ap-

pearing in Φ. Such a triple is a *positive* instance of 2-QBF iff $\exists \vec{a} \in 2^A$ such that $\forall \vec{b} \in 2^B$, $(\vec{a}, \vec{b}) \models \Phi$. A characterization of the positive instances of the canonical Π_2^p-complete problem 2-$\overline{\text{QBF}}$ is obtained by reversing the quantifiers, namely: $\forall \vec{a} \in 2^A \ \exists \vec{b} \in 2^B$ such that $(\vec{a}, \vec{b}) \models \Phi$. Finally, an instance of the canonical Π_3^p-complete problem 3-$\overline{\text{QBF}}$ is a 4-uple $\langle A, B, C, \Phi \rangle$, where $\{A, B, C\}$ is a partition of $Var(\Phi)$, and the positive ones are such that $\forall \vec{a} \in 2^A \ \exists \vec{b} \in 2^B \ \forall \vec{c} \in 2^C$, $(\vec{a}, \vec{b}, \vec{c}) \models \Phi$. The following easy result shows that positive instances of 2-QBF and of 2-$\overline{\text{QBF}}$ can be characterized using prime implicants modulo a formula:

Proposition 1 (2-QBF and prime implicants)

- $\langle A, B, \Phi \rangle$ *is a positive instance of* 2-QBF *if and only if* $PI^A(\Phi) \neq \emptyset$.
- $\langle A, B, \Phi \rangle$ *is a positive instance of* 2-$\overline{\text{QBF}}$ *if and only if* $PI^A(\neg\Phi) = \emptyset$.

Accordingly, with $\Sigma = \{a \vee b, (a \wedge c) \Rightarrow e, d \Rightarrow e\}$, $\langle \{b, c, e\}, \{a, d\}, \Sigma \rangle$ is a positive instance of 2-QBF (because $PI_{\top}^{\{b,c,e\}}(\Sigma) = \{b \wedge e\}$) while $\langle \{a, b, c\}, \{d, e\}, \Sigma \rangle$ is not (because $PI_{\top}^{\{a,b,c\}}(\Sigma) = \emptyset$). $\langle \{a\}, \{b, c, d, e\}, \Sigma \rangle$ is a positive instance of 2-$\overline{\text{QBF}}$ while $\langle \{a, b\}, \{c, d, e\}, \Sigma \rangle$ is not. Moreover, $\langle \{a\}, \{b, c, e\}, \{d\}, \Sigma \rangle$ is a positive instance of 3-$\overline{\text{QBF}}$ while $\langle \{a\}, \{b, c\}, \{d, e\}, \Sigma \rangle$ is not.

2.3 Dependency-based update and the MPMA

We conclude this section with some basics about dependency-based update [16] and the Modified Possible Models Approach (MPMA) [10]. Here, the syntactic presentations of updated knowledge bases do not matter, hence we define them model-theoretically by sets of models. Let $P \subseteq PS$, $\vec{\omega}$ a world and α a formula. The *MPMA-update* [10] of $\vec{\omega}$ by α w.r.t. P, denoted by $\vec{\omega} \star^P \alpha$, is the set of worlds $\vec{\omega}'$ s.t. $\vec{\omega}' \models \alpha$ and for every propositional variable v not in P, $\vec{\omega}$ and $\vec{\omega}'$ assign the same truth value to v. If KB is a formula from $PROP_{PS}$, then the MPMA-update $KB \star^P \alpha$ of KB by α w.r.t. P is defined by $KB \star^P \alpha = \bigcup_{\vec{\omega} \models KB} (\vec{\omega} \star^P \alpha)$. The set of variables P is allowed to vary. At least four specific choices are worth considering for P:

1. one extreme case is $P = PS$, which leads to forgetting everything about KB. Namely, $KB \star^{PS} \alpha \equiv \alpha$;
2. the other extreme case is $P = \emptyset$, which comes down to classical conjunction: $KB \star^{\emptyset} \alpha \equiv KB \wedge \alpha$;
3. as proposed by Herzig [16], P may be the set of variables $Var(\alpha)$ mentioned in α – this appears reasonable because variables not appearing in α can be considered having nothing to do with α – thus it is not necessary to forget about them before adding α;
4. however, it might be the case that some variables appear (syntactically) in α while being (semantically) irrelevant to α. Doherty et al. [10] propose the choice of $P = DepVar(\alpha)$, namely the set of variables α is dependent on, defined by: $p \in DepVar(\alpha)$ iff p occurs in any formula α' logically equivalent to α, or, equivalently, $p \in DepVar(\alpha)$ iff $\alpha[p \leftarrow \top]$ is not equivalent to $\alpha[p \leftarrow \bot]$ [10] [22] [21] [1].

Doherty et al. [10] give the following syntactical characterization of the MPMA: if $P = \{p_1, \ldots p_n\}$, then $\exists p_i \alpha = \alpha[p_i \leftarrow \top] \vee \alpha[p_i \leftarrow \bot]$ and $\exists P.KB$ stands for $\exists p_1 \ldots \exists p_n KB$. The following holds: $KB \star^P \alpha \equiv \alpha \wedge \exists P.KB$. Thus, $KB \star^P \alpha$ is equivalent to a (simple) quantified boolean formula (with some free variables!). Such a formula can be turned into a classical one using variable elimination algorithms (e.g., directional resolution [9]), but there is no guarantee that the size of the resulting formula is polynomial in the input size (and the existence of such a rewriting is unlikely since it would make the polynomial hierarchy to collapse at the first level [21]). Interestingly, in the case where $P \supseteq DepVar(\alpha)$ (e.g., in cases 1., 3. or 4. above), the above characterization of MPMA-update through quantified boolean formula shows that $KB \star^P \alpha$ is consistent as soon as KB is consistent and α is consistent.

3 Qualitative one-stage decision problems

3.1 Knowledge, goals, observations, and decisions

A qualitative decision problem intuitively consists of a partition of the variables of interest between decision and state variables (where each state variable is either observable or unobservable), a knowledge base K describing the initial state of the world, a set of constraints C restricting the different decisions available to the agent, the description of the effects of decisions EFF, and a set of goals G.

[1] Contrariwise to Doherty et al. [10], we do not think that such a characterization is impractical from a computational point of view. Indeed, computing $DepVar(\alpha)$ can be achieved through "only" a linear number of calls to an NP oracle. Contrastingly, the characterization presented in [10] requires the computation of the set of prime implicants of α, whose cardinality is exponential in the size of α in the worst case.

Definition 1 (qualitative decision problem) *A* qualitative decision problem (QDP) *is a 7-uple* $\mathcal{P} = \langle DecVar, ObsVar, UnobsVar, K, C, G, EFF \rangle$ *where:*

- $\{DecVar, ObsVar, UnobsVar\}$ *is a partition of* PS. *Variables of* $DecVar$ *(resp.* $ObsVar$, $UnobsVar$*) are* decision variables *(resp.* observable state variables, unobservable state variables*).* $StateVar = ObsVar \cup UnobsVar$ *is the set of* state variables.
- K *and* G *are finite consistent sets of formulas on* $StateVar$. C *is a consistent set of formulas on* $DecVar$.
- EFF *is a finite set of triples* $\langle \varphi, \delta, \psi \rangle$ *where* φ *and* ψ *are consistent formulas on* $StateVar$ *and* δ *is a consistent formula on* $DecVar$.

Decision variables are under the control of the agent (the latter has the power to fix their truth value). Such variables are sometimes called *controllable* [1] [23]. A complete assignment $\vec{d}$ of all decision variables is called a *decision*. The decision space 2^{DecVar} is denoted by D.

State variables describe the state of the world (before and after the decision is taken). They are not *directly* under the control of the agent but some of them may be influenced by the agent. They are partitioned into two subclasses: *observable (state) variables* are observed, i.e., their truth value is known before the agent acts, whereas the truth value of *unobservable (state) variables* is not directly observable by the agent. A complete (resp. partial) assignment $\vec{o}$ of observable variables is called a complete (resp. partial) *observation*. The set of all (complete or partial) observations (*observation space*) is denoted by O. $\vec{u}$ denotes a complete assignment of unobservable variables; the set of all such assignments is denoted by U. If $\vec{o}$ is a complete observation and $\vec{u}$ is a complete assignment of unobservable variables, then $\vec{s} = (\vec{o}, \vec{u})$ is a complete assignment of all state variables. S denotes the set of all such assignments (called *states*).

K, bearing on state variables only, describes the *initial state* of the world. Of course, in the general case, only partial knowledge about this state is available (i.e., K may easily have several models on $StateVar$).

EFF describes the *effects of decisions*. Each triple $\langle \varphi, \delta, \psi \rangle$ of EFF has the following meaning: "if the initial state (before the decision is taken) satisfies φ and the decision undertaken satisfies δ, then the resulting state satisfies ψ". Clearly, these triples used for representing effects of actions are similar to the transitions used by Cordier and Siegel in [7].

Formulas of C are (physical) *constraints* restricting the allowed combination of decision variables: if such a combination $\vec{d}$ does not satisfy C, then the corresponding decision is not available to the agent[2].

Formulas of G are *goals*. They are specified by the agent and express his/her preferences about the consequences of his/her decisions. They involve state variables only – without loss of generality since decision variables upon which the agent has preferences *can be duplicated* by as many state variables, through the expansion of EFF with additional effects rules of the form $\langle \top, d, d_{obs} \rangle$ and $\langle \top, \neg d, \neg d_{obs} \rangle$ for every $d \in DecVar$.

Three particular cases are of interest regarding the variables:

- $ObsVar = \emptyset$: all state variables are unobservable and $\mathcal{P}$ is a QDP under *unobservability* (UQDP).
- $UnobsVar = \emptyset$: all state variables are observable and $\mathcal{P}$ is a QDP under *full observability* (FQDP).
- $UnobsVar = \emptyset$ and moreover there is only one possible state consistent with K: the initial state is completely known in advance (*off-line*) – to be distinguished from the more general case of full observability where the initial state is fully observed *on-line*. $\mathcal{P}$ is a QDP under *complete initial knowledge*.

We consider particular cases regarding constraints:

- *Full executability* corresponds to $C = \emptyset$ and means that any combination of decision variables is allowed, which makes $2^{|DecVar|}$ available decisions.
- *Atomicity* corresponds to the case where C contains $\neg(d \wedge d')$ for all $d, d' \neq d \in DecVar$. Decisions correspond then to atomic actions (which is the classical assumption in planning and in Markov Decision Processes). Under atomicity, only $|DecVar| + 1$ decisions are available: every decision that satisfies more than one $d_i \in DecVar$ is not consistent with C.

We also consider particular cases regarding the effects of decisions:

[2] More generally, we could have allowed *conditional* constraints, i.e., pairs $\langle \varphi, \gamma \rangle$ (where φ involves state variables) meaning "if the initial state satisfies φ, then the decision taken must satisfy γ". For the sake of simplicity, we omit this possibility.

- STRIPS: (i) for each $\langle \varphi, \delta, \psi \rangle$ in EFF, φ and ψ are terms on $StateVar$; (ii) K is a complete term on $StateVar$ (i.e., each variable of $StateVar$ occurs once in K, either positively or negatively); (iii) G is a term on $StateVar$; (iv) EFF satisfies atomicity. Clearly enough, due to (ii), only full observability makes sense under the STRIPS restriction.
- STRIPS$^+$: (i) as above and nothing else (K and G are any formulas on $StateVar$ and atomicity is not required).
- *qualitative influence diagrams* (QID): for each $\langle \varphi, \delta, \psi \rangle$ in EFF, ψ is a literal [3].

For the sake of simplicity, ramifications are handled explicitely in this paper. They could be handled automatically using either explicit dependencies as in [16] or integrity constraints as in [10]. We omit this in order to simplify the exposition.

Example 1 (aunt Agatha's living-room, revisited).

$ObsVar$	=	{cold, summer, stuffy};
$UnobsVar$	=	$\emptyset$;
$DecVar$	=	{heating, (window-)open};
K	=	{summer $\Rightarrow \neg$cold};
C	=	$\emptyset$;
EFF	=	{ $\langle$cold, heating $\wedge \neg$open, $\neg$cold$\rangle$, $\langle \top$, open, $\neg$stuffy$\rangle$, $\langle \neg$summer, open, cold$\rangle$};
G	=	{$\neg$cold, $\neg$stuffy}.

K and G do not need any explanation; let us briefly explain the effects items:

- $\langle$cold, heating $\wedge \neg$open, $\neg$cold$\rangle$ means that if the room is initially cold and if the decision taken includes (among other things) putting the heating on and closing the window, then in the resulting state the room will not remain cold.
- $\langle \top$, open, $\neg$stuffy$\rangle$ means that opening the window makes the room non-stuffy. We do not assume here that closing the window of a non-stuffy room does make it stuffy. If we want to represent the fact that closing the window *may* make the room stuffy (and may not), then one way to do it is adding the effect rule $\langle \top$, $\neg$open, stuffy $\vee$lucky$\rangle$, where "lucky" is a new state variable, or alternatively, the effect rule $\langle$lucky, $\neg$open, $\neg$stuffy$\rangle$, where "lucky" is a new *unobservable* state variable. Another way would consist in adding an explicit dependency (as in [16]) telling that closing the window influences the variable "stuffy".
- $\langle \neg$summer, open, cold$\rangle$ means that if it is not summertime, opening the window will make the room cold.

[3] This terminology comes, of course, from influence diagrams [18] where a probability for each state variable, conditioned by the values of the decision and state variables it depends on, is specified. Imposing that only literals appear in the consequence part of decision effects correspond thus to the description of such a conditional probability in a degenerate influence diagram where variables are binary and probabilities are either 0 or 1.

3.2 States, decisions and consequences

Let us now define the following notions of possible observation, possible state, available decision and consequence of a decision on a state given a QDP $\mathcal{P}$.

Definition 2 (possible observations)
An observation $\vec{o}$ is possible *(w.r.t. $\mathcal{P}$) iff $K \wedge \vec{o}$ is consistent. The set of all possible observations is denoted by $PossObs(\mathcal{P})$.*

Definition 3 (possible states)
A state $\vec{s}$ is possible *(w.r.t. $\mathcal{P}$) iff $\vec{s} \models K$. The set of all possible states is denoted by $PossStates(\mathcal{P})$.*

Definition 4 (available decisions) *A decision $\vec{d}$ is* available *(w.r.t. $\mathcal{P}$) iff $\vec{d} \models C$. The set of all available decisions is denoted by $AvDec(\mathcal{P})$.*

When no ambiguity is possible about the underlying QDP $\mathcal{P}$, the simplified notations $PossObs$, $PossStates$, $AvDec$ will be used instead of, respectively, $PossObs(\mathcal{P})$, $PossStates(\mathcal{P})$, $AvDec(\mathcal{P})$.

Definition 5 (consequences of $\vec{d}$ on $\vec{s}$)
Given a QDP $\mathcal{P}$, let $\vec{d} \in AvDec$ and $\vec{s} \in PossStates$.

- $Collect(\vec{d}, \vec{s}) = \bigwedge \left\{ \psi \;\middle|\; \begin{array}{l} \langle \varphi, \delta, \psi \rangle \in EFF \text{ such} \\ \text{that } \vec{s} \models \varphi \text{ and } \vec{d} \models \delta \end{array} \right\}$;
- $Cons(\vec{d}, \vec{s}) = \vec{s} \star^{P(\vec{d},\vec{s})} Collect(\vec{d}, \vec{s})$ [4] *(here, $Cons(\vec{d}, \vec{s})$ is identified with the set of its models).*

Thus, the consequence of a decision $\vec{d}$ on a state $\vec{s}$ is computed in two steps: first, $Collect(\vec{d}, \vec{s})$ filters all "active" effect rules and gathers their consequence parts; an effect rule $\langle \varphi, \delta, \psi \rangle$ is active if its precondition φ is satisfied by $\vec{s}$ and its decision part δ is satisfied by

[4] To be more rigorous we should use the notation $Cons_{P(\vec{d},\vec{s})}(\vec{d}, \vec{s})$ instead of $Cons(\vec{d}, \vec{s})$. In order to avoid heavy notations, we omit the subscript $P(\vec{d}, \vec{s})$.

$\vec{d}$. Then, $\vec{s}$ is updated with the formula resulting from the preceding step, using the MPMA. We note that the parameter $P(\vec{d}, \vec{s})$ has been left unspecified. The following four choices are worth considering (the first three choices are also commented in the introductory section):

1. $P(\vec{d}, \vec{s}) = PS$, whatever $\vec{s}$ and $\vec{d}$.
2. $P(\vec{d}, \vec{s}) = Var(Collect(\vec{d}, \vec{s}))$.
3. $P(\vec{d}, \vec{s}) = DepVar(Collect(\vec{d}, \vec{s}))$. As shown in [10], this choice, namely the set of variables on which $Collect(\vec{d}, \vec{s})$ is dependent, works well in practice. Unfortunately, computing $DepVar(Collect(\vec{d}, \vec{s}))$ is NP-hard [22] and this task can not be preprocessed because there are exponentially many formulas $Collect(\vec{d}, \vec{s})$ in the worst case.
4. $P(\vec{d}, \vec{s}) = \bigcup\{DepVar(\psi) | \langle \varphi, \delta, \psi \rangle \in EFF, \vec{s} \models \varphi, \vec{d} \models \delta\}$. This amounts to a preprocessing phase where a dependency set $DepVar(\psi)$ is computed *for every effect rule* $\langle \varphi, \delta, \psi \rangle$ in EFF, and a linear-time postprocessing phase consisting in taking the union of such variables for all fired effect rules. It is not hard to see that in this case $P(\vec{d}, \vec{s})$ is an *upper approximation* of $DepVar(Collect(\vec{d}, \vec{s}))$, i.e., we have $P(\vec{d}, \vec{s}) \supseteq DepVar(Collect(\vec{d}, \vec{s}))$. The equality does not hold in the general case: for instance, if $EFF = \{\langle \top, x, a \vee b \rangle, \langle \top, y, \neg a \vee b \}$, $\vec{s} = a \wedge b$, $\vec{d} = x \wedge y\}$, then $Collect(\vec{d}, \vec{s}) \equiv b$ and $DepVar(Collect(\vec{d}, \vec{s})) = \{b\}$ while $DepVar(a \vee b) \cup DepVar(\neg a \vee b) = \{a, b\}$. However, whether the best choice here for $P(\vec{d}, \vec{s})$ is $\{a, b\}$ or $\{b\}$ is unclear. If $P(\vec{d}, \vec{s}) = \{b\}$, then $\vec{s} \star^{\{b\}} Collect(\vec{d}, \vec{s}) = \vec{s} \star^{\{b\}} b = a \wedge b$, whereas if $P(\vec{d}, \vec{s}) = \{a, b\}$, then $\vec{s} \star^{\{a,b\}} b = b$. Which one is intuitively more satisfactory is left for further research.

Now, in the general case the agent only has a *partial knowledge* about the real world, which leads us to defining the consequence of a decision on an observation.

Definition 6 (consequences of $\vec{d}$ on $\vec{o}$)
Given a QDP $\mathcal{P}$, for any $\vec{d} \in AvDec$ and $\vec{o} \in PossObs$, $Cons(\vec{d}, \vec{o})$ is defined model-theoretically by the set of models $\bigcup\{Cons(\vec{d}, \vec{s}) \mid \vec{s} \models \vec{o}\}$.

Example 1 (continued)
Let us make any of the three assumptions 2, 3, 4 for the choice of $P(\vec{d}, \vec{s})$, and let $\vec{o}$ = cold, $\vec{d}$ = heating $\wedge\neg$open; $\vec{d'}$ = heating $\wedge$open. We have:
- $Cons(\vec{d}, \vec{o}) \equiv$ cold $\wedge\neg$summer.
- $Cons(\vec{d'}, \vec{o}) \equiv$ cold $\wedge\neg$summer $\wedge\neg$stuffy.

For $\vec{d}$, the two consequent states come respectively from the two possible states $\vec{s}$ = (cold, $\neg$summer, stuffy) and $\vec{s'}$ = (cold, $\neg$summer, $\neg$stuffy). Should we have done the choice $P(\vec{d}, \vec{s}) = PS$ then we would have got $Cons(\vec{d}, \vec{o}) \equiv$ cold, because nothing tells *explicitely* that summertime does not come in the meanwhile, even if nothing tells the contrary: implicit persistence is not handled.

3.3 Consistency and determinism

Some global properties can be expected from a QDP in order to ensure that its specification is correct. Especially, it can be expected that each time an observation is possible and a decision is available, the decision can be taken without producing an inconsistency. More formally:

Definition 7 (consistency)
A QDP $\mathcal{P}$ is consistent *iff $\forall \vec{s} \in PossStates$, $\forall \vec{d} \in AvDec$, $Collect(\vec{d}, \vec{s})$ is consistent.*

It is easy to check that the QDP given at Example 1 is consistent.

Consistency guarantees that the description of causal laws (between initial states, decisions and consequent states) is well-founded, i.e., it allows at least one possible consequence for any available decision and any possible initial state[5]. *From now on, we assume that all the QDPs considered are consistent.*

It is important to remark that consistency implies that $Cons(\vec{d}, \vec{s}) \neq \emptyset$ as soon as $P(\vec{d}, \vec{s}) \supseteq DepVar(Collect(\vec{d}, \vec{s}))$, (consequence of Theorem 2 in [10]) - and our four possible choices for $P(\vec{d}, \vec{s})$ are all supersets of $DepVar(Collect(\vec{d}, \vec{s}))$.

QDP CONSISTENCY is the decision problem that consists in determining whether a given QDP is consistent. As the following proposition shows it, checking consistency is computationally a heavy task in the general case.

Proposition 2 (QDP CONSISTENCY)
The complexity of QDP CONSISTENCY *is reported in the following two tables:*

[5] Actually, to a QDP can be associated a causal structure similar to a causal network [8] augmented with decision variables and a goal variable. Our definition of consistency has some similarities (though is not equivalent to) causal independence.

	general case
general case	Π_2^p-complete
QID	coNP-complete
STRIPS$^+$	coNP-complete

	+ atomicity	+ full executability
general case	Π_2^p-complete	Π_2^p-complete
QID	coNP-complete	coNP-complete
STRIPS$^+$	coNP-complete	coNP-complete
STRIPS	in P	-

Another valuable property a QDP can exhibit is determinism:

Definition 8 (determinism)
A QDP $\mathcal{P}$ is deterministic *iff* $\forall \vec{s} \in PossStates$, $\forall \vec{d} \in AvDec$, $Cons(\vec{d}, \vec{s})$ *is a singleton.*

Determinism guarantees that given a possible complete initial state and an available decision, all the state variables are determined once the decision has been made on that state. Clearly enough, determinism entails consistency. Importantly, whether a QDP is deterministic or not *depends on the choice for the set $P(\vec{d}, \vec{s})$ used in the update operation* (contrariwise to consistency).

For instance, in Example 1, the QDP is deterministic with $P(\vec{d}, \vec{s}) = DepVar(Collect(\vec{d}, \vec{s}))$ but nondeterministic if $P(\vec{d}, \vec{s}) = PS$.

Interestingly, when the choice for $P(\vec{d}, \vec{s})$ is 1., 2. or 4., determinism is easier to check than consistency. Let QDP DETERMINISM the problem of deciding whether a given QDP is deterministic or not; we have identified the following complexity result:

Proposition 3 (QDP DETERMINISM)
When $P(\vec{d}, \vec{s})$ is computed in polynomial time, QDP DETERMINISM *is* coNP*-complete.*

Interestingly, under the QID assumption or the STRIPS$^+$ assumption, consistency implies determinism when $P(\vec{d}, \vec{s}) = DepVar(Collect(\vec{d}, \vec{s}))$.

Proposition 4 *Let $\mathcal{P}$ be a QDP whose syntax satisfies the* QID ASSUMPTION *or the* STRIPS$^+$ *assumption. Then, if the choice for $P(\vec{d}, \vec{s})$ is 2., 3. or 4.*[6]*, the consistency of $\mathcal{P}$ implies its determinism.*

Indeed, under the QID assumption when $\mathcal{P}$ is consistent, the truth value of each state variable is determined in $Cons(\vec{d}, \vec{s})$; since this is true for each available $\vec{d}$ and each possible $\vec{s}$, $\mathcal{P}$ is deterministic.

[6]Note that they coincide under the QID assumption when $\mathcal{P}$ is consistent.

4 Solving a QDP

Let us now explain what solving a QDP means.

4.1 Observation covering

Solving a QDP consists in determining, given any possible observation, a decision covering it, i.e., that can ensure the goal to be satisfied when the decision is taken *whatever the values of the unobservable variables.*

This calls for the following notions:

Definition 9 (cover, coverable observation)
Let $\mathcal{P}$ be a consistent QDP. Let $\vec{o} \in PossObs$ and $\vec{d} \in AvDec$. $\vec{d}$ covers *$\vec{o}$ iff $Cons(\vec{d}, \vec{o}) \models G$. When there exists a $\vec{d}$ covering $\vec{o}$, $\vec{o}$ is said* coverable.

A rewriting of the definitions leads to the following characterization for observation covering: $\vec{o}$ is coverable iff $\exists \vec{d} \in D$ s.t. $\forall \vec{s} \models K \wedge \vec{o}$, $\forall \vec{s'} \in Cons(\vec{d}, \vec{s})$, $(\vec{d}, \vec{s'}) \models C \wedge G$. Clearly, this looks like a 2-QBF problem (this is completely clear when we choose $P(\vec{d}, \vec{s}) = PS$). This enables determining the complexity of OBSERVATION COVERING:

Proposition 5 (OBSERVATION COVERING)
Let $\mathcal{P}$ be a consistent QDP:

1. *Determining whether a given available decision $\vec{d}$ covers a given possible observation $\vec{o}$ is* coNP*-complete*[7].
2. *Determining whether a given possible observation $\vec{o}$ is coverable is Σ_2^p-complete.*

OBSERVATION COVERING still remains coNP-complete under the QID assumption or the STRIPS$^+$ assumption, whatever the choice for $P(\vec{d}, \vec{s})$ (among 1., 2., 3., or 4.), even in the restricted case of unobservability. Moreover, OBSERVATION COVERING remains coNP-complete in the general case under full observability whatever the choice for $P(\vec{d}, \vec{s})$. Additionally, OBSERVATION COVERING remains coNP-complete under the QID or the STRIPS$^+$) assumption in the restricted case of full observability when the choice for $P(\vec{d}, \vec{s})$ is 1. Lastly, in the case of full observability under the QID or the STRIPS$^+$) assumption, and when thc choice for $P(\vec{d}, \vec{s})$ is 2,3 or 4; under the same conditions, the membership to P holds for the choice 1 if G is under conjunctive normal form.

[7]We have also identified the complexity of MPMA as an extra-result: while computing $DepVar(\alpha)$ is already NP-hard, determining if $KB \star^{DepVar(\alpha)} \alpha \models \psi$ holds (given KB, α and ψ) is "only" coNP-complete.

Interestingly, in some cases the notion of cover can be characterized using prime implicants, taking advantage of the connection between 2-QBF, 2-$\overline{\text{QBF}}$ problems and prime implicants given in Proposition 1. Before, we need to rewrite the MPMA into classical deduction (in the same spirit as Lemma 1 in [24] for dependence-based update [16] [17]). The idea consists in (i) duplicating state variables: to each state variable $v \in StateVar$ we associate two state variables v_t and v_{t+1} which refer respectively the the state of the world *before* and *after* the update; note that decision variables need not being duplicated; (ii) translating the effect rules $\langle \varphi, \delta, \psi \rangle$ following this intuitive formulation: if the precondition φ is true at time t (before the update) and if the decision taken satisfies the firing condition δ, then the postcondition ψ is true at time $t+1$ (after the update); (iii) write a *completion* (frame axiom) ensuring that variables not concerned by the update persist. However, in the general case, step (iii) is problematic, because the variables that are not concerned with an update *depend both on the decision taken and on the initial state*, and there are exponentially many possible initial states and available decisions in the worst case. In order to cope with this problem, we proceed the other way round: each state variable is considered separately, and we determine a sufficient condition under which it is not concerned by the update. Of course, this completion depends crucially on the choice made for $P(\vec{d}, \vec{s})$. We will see that if the choice for $P(\vec{d}, \vec{s})$ is 1., 2. or 4., then the sufficient condition found is necessary; only for the case of choice 3., this sufficient condition is not necessary.

For any formula F of $PROP_{PS}$, F_t is obtained from F by replacing each state variable v occurring in it by v_t, and similarly for F_{t+1}. For a world $\vec{x}$, we note $\vec{x}_t$ and $\vec{x}_{t+1}$ the corresponding worlds after time indexing.

Definition 10 (persistence completion)
Let $\mathcal{P}$ be a QDP. We define successively:

- *for any effect rule $R = \langle \varphi, \delta, \psi \rangle \in EFF$, the translation of R is $Trans(R) = (\varphi_t \wedge \delta) \Rightarrow \psi_{t+1}$;*
- *for any state variable $v \in StateVar$, the* potentially dependent effect rules *of v is defined by:* $PDR(v) = \{\langle \varphi, \delta, \psi \rangle \in EFF \mid v \in DepVar(\psi)\}$;
- *for any state variable $v \in StateVar$, the* persistence completion *of v is defined by:* $Comp(v) = [\bigwedge_{\langle \varphi, \delta, \psi \rangle \in PDR(v)} \neg(\varphi_t \wedge \delta)] \Rightarrow (v_t \Leftrightarrow v_{t+1})$;
- *finally, the translation of EFF is defined by:* $Trans(EFF) = (\bigwedge_{R \in EFF} Trans(R)) \wedge (\bigwedge_{v \in StateVar} Comp(v))$.

Proposition 6 (cover and prime implicants)
For any $\vec{d} \in AvDec$ and $\vec{o} \in PossObs$, $\vec{d}$ covers $\vec{o}$ if and only if there is a γ in $PI^{ObsVar_t \cup DecVar}_{K_t \wedge Trans(EFF)}(C \wedge G_{t+1})$ such that $(\vec{d}, \vec{o}_t) \models \gamma$.

Example 1 (continued) We have

$PI^{ObsVar_t \cup DecVar}_{K_t \wedge Trans(EFF)}(C \wedge G_{t+1})$
$= \{ (\neg cold_t \wedge \neg stuffy_t \wedge \neg open), (\neg stuffy_t \wedge heating \wedge \neg open), (summer_t \wedge \neg stuffy_t), (summer_t \wedge open) \}.$

Accordingly, with $\vec{d}'' = \neg heating \wedge \neg open$, we have

$PI^{ObsVar_t}_{K_t \wedge Trans(EFF) \wedge \vec{d}''}(C \wedge G_{t+1})$
$= \{(\neg cold_t \wedge \neg stuffy_t), (summer_t \wedge \neg stuffy_t)\}$

We can check that $\vec{d}''$ covers both $\neg cold \wedge \neg stuffy$ and $summer \wedge \neg stuffy$.

Note that if we replace $DepVar(\psi)$ in the definition of $PDR(v)$ by $Var(\psi)$, then we get a similar result *for choice 2.*. And lastly, if we simply ignore persistence, i.e., if we set $Comp(v) = \emptyset$ for all $v \in StateVar$, then we get a similar result *for choice 1.* No similar result can be obtained for choice 3. Accordingly, the previous proposition shows that if the choice for $P(\vec{d}, \vec{s})$ is 1., 2. or 4., then deciding whether any observation $\vec{o}$ is covered by a given decision $\vec{d}$ is easy, once $PI^{ObsVar_t \cup DecVar}_{K_t \wedge Trans(EFF)}(C \wedge G_{t+1})$ has been computed (this set can be viewed as a compilation for the OBSERVATION COVERING problem). It also shows that OBSERVATION COVERING can be interpreted as an abduction problem: $\vec{d}$ covers $\vec{o}$ iff $\vec{o}_t$ implies a minimal (for $\subseteq$) abductive explanation for $C \wedge G_{t+1}$ w.r.t. $K_t \wedge Trans(EFF) \wedge \vec{d}$, where the set of possible individual hypotheses is $ObsVar_t$ (see [13] for more details).

4.2 Policies

A *policy* for a QDP associates available decisions to possible observations. The resolution of a QDP consists ideally in finding out a policy π mapping every possible observation $\vec{o}$ to an available decision $\vec{d} = \pi(\vec{o})$ that covers it. In this situation, for any possible $\vec{o}$, applying $\pi(\vec{o})$ ensures that a goal state will be reached. Such a policy is called a *solution policy* for $\mathcal{P}$.

Definition 11 (policies)
A partial policy *π for a QDP $\mathcal{P}$ is a mapping from $Dom(\pi) \subseteq PossObs(\mathcal{P})$ to $AvDec(\mathcal{P})$.*
It is sound *iff $\forall \vec{o} \in Dom(\pi)$, $\pi(\vec{o})$ covers $\vec{o}$.*
It is complete *iff $Dom(\pi) = PossObs(\mathcal{P})$.*
A solution policy *is a sound and complete policy.*

Specifying a policy explicitely (by listing all observations of $Dom(\pi)$ together with their assigned decision)

is exponentially long in the general case – except in the case where $ObsVar$ has a bounded cardinality – since there may be up to $2^{|ObsVar|}$ observations. So as to deal with this problem, we consider (like in [2]) structured descriptions of policies where decisions are assigned not to complete observations but to (typically partial) observations representing arbitrarily large sets of complete observations.

Definition 12 (structured descriptions)
A structured description of a partial policy is a finite set of pairs $\sigma = \{\langle \vec{o}_i, \vec{d}_i \rangle, i = 1..r\}$.

$\vec{o}_i$ is the *application context* of $\vec{d}_i$. The partial policy π induced by σ is defined in a natural way [8]. Let $\vec{o}$ be a complete observation, then:

if there is a $\langle \vec{o}_i, \vec{d}_i \rangle$ in σ such that $\vec{o} \models \vec{o}_i$
then $\pi(\vec{o}) := \vec{d}_i$ where i is the *first* index of $1 .. r$ such that $\vec{o} \models \vec{o}_i$
else $\pi(\vec{o})$ is undefined.

Note that we have $Dom(\pi) = \bigvee_{i=1}^{r} \vec{o}_i$.

Clearly enough, OBSERVATION COVERING from a QDP given a structured description σ of a policy for it is in P: it is sufficient to successively consider the application contexts of σ and to check whether one of them is s.t. $\vec{o} \models \vec{o}_i$. If so, $\vec{o}$ is coverable and $\vec{d}_i$ is a covering decision, otherwise $\vec{o}$ is not coverable.

Since it is often the case that there is no solution policy (because some observation is not coverable), we consider now the weaker notion of *maximal sound policy*, i.e., a policy in which every coverable observation is covered (clearly, every solution policy is a maximal sound policy).

Definition 13 (maximal sound policies)
A maximal sound policy π *is a sound policy such that there is no sound policy* π' *that satisfies* $Dom(\pi') \supset Dom(\pi)$.

Example 1 (continued)
With $P(\vec{d}, \vec{s}) = DepVar(Collect(\vec{d}, \vec{s}))$, there is no solution policy for $\mathcal{P}$, because the complete observations (cold ∧¬summer ∧stuffy) and (¬cold ∧¬summer ∧stuffy) cannot be covered. Of course, every partial observation that is extracted from any of these two complete observations is not coverable as well. Here is a maximal sound policy under structured form: {⟨¬summer ∧¬stuffy, heating ∧¬open⟩, ⟨summer, heating ∧open ⟩}. Another maximal sound policy under structured form is: {⟨¬stuffy, heating ∧¬open⟩, ⟨summer ∧stuffy, heating ∧open ⟩}.

We now study the complexity of QDP POLICY EXISTENCE that is the decision problem consisting in determining whether a solution policy exists for a QDP; we make the following parameters vary:

1. observability (partial, full or none);
2. particular assumptions on K, C and EFF.

The following results hold whatever the choice for the parameter $P(\vec{d}, \vec{s})$ (among 1., 2., 3. or 4.).

Proposition 7 (QDP POLICY EXISTENCE)
The complexity of QDP POLICY EXISTENCE *is reported in the following two tables.*

	partially observable
general case	Π_3^p-complete
QID	Π_3^p-complete
STRIPS$^+$	Π_3^p-complete

	fully observable	unobservable
general case	Π_3^p-complete	Σ_2^p-complete
QID	Π_2^p-complete	Σ_2^p-complete
STRIPS$^+$	Π_2^p-complete	Σ_2^p-complete
STRIPS	in P	-

Clearly enough, under unobservability, a solution policy exists iff the empty observation is coverable (i.e., we can find a unique decision that, whatever the initial state, enables to reach a goal state if taken).

It can be shown that solving a QDP amounts at "solving" a quantified boolean formula. Since the formulation is heavy in the general case, we show it in the specific case where $P(\vec{d}, \vec{s}) = PS$. Let $E = \{(\varphi_t \wedge \delta) \Rightarrow \psi_{t+1} \mid \langle \varphi, \delta, \psi \rangle \in EFF\}$. Then $\mathcal{P}$ has a solution policy iff

$$\forall \vec{o}_t \in O_t \; \exists \vec{d} \in D \; \forall \vec{s}_t \models \vec{o}_t \; \forall \vec{s}_{t+1} \in S_{t+1}, \; (\vec{s}_t, \vec{d}, \vec{s}_{t+1}) \models (K_t \wedge E) \Rightarrow (C \wedge G_{t+1})$$

We recognize here an instance of 3-$\overline{\text{QBF}}$. In the particular case of unobservability, it reduces to a 2-QBF problem.

4.3 Computing policies

Computing a policy can be viewed as a form of *compilation* for OBSERVATION COVERING: decisions are associated to observations during an off-line compilation

[8] Since the $\vec{o}_i$ are not required to be mutually inconsistent, it may be the case that more than one $\vec{o}_i$ is satisfied by $\vec{o}$, which means that more than one of the $\vec{d}_i$ covers $\vec{o}$: in this case, any of these decisions covering $\vec{o}$ can be picked up – here we choose arbitrarily to select the one appearing first in σ.

phase, then when the values of the observable variables are known, finding out a decision covering the observation can be achieved on-line in a more efficient way than it would be the case, would this decision be computed from scratch. The time spent in computing a policy can be compensated by the time gained when the same qualitative decision problem is to be frequently repeated with different input observations.

In the following, we present a method for computing structured descriptions of maximal sound policies, based on the computation of prime implicants. Decisions are considered in a successive way, and for each selected decision, an application context is computed. Application contexts are computed for successive decisions $\vec{d}$, and the negation of each of these applications contexts is then added to K so as not to consider the context again.

Input $\mathcal{P}$ (a QDP)
Output σ (a structured description of a partial policy)
Begin
  $\sigma := \emptyset$;
  While $K \wedge G \wedge C$ is consistent **do**
    i. find $\vec{d}$ not appearing in σ and such that $K \wedge G \wedge C \wedge \vec{d}$ is consistent;
    ii. $L := PI^{ObsVar_t}_{K_t \wedge Trans(EFF) \wedge \vec{d}}(C \wedge G_{t+1})$;
    iii. **if** L is not empty **then**
      for each γ_t in L **do**
        add $\langle \gamma, \vec{d} \rangle$ to σ;
        $K \leftarrow K \wedge \neg\gamma$;
      end for
    end if
  end while
  Return σ
End.

At line (i), the search for $\vec{d}$ can be done by searching first for a model of $K \wedge G \wedge C$ and then projecting it on $DecVar$.

At line (ii), $PI^{ObsVar_t}_{K_t \wedge Trans(EFF) \wedge \vec{d}}(C \wedge G_{t+1})$ can be computed in an incremental way. In practice, such prime implicants are derived through the generation of the duals, i.e., prime implicates modulo a formula, for which sophisticated algorithms exist (e.g., [19]). Such algorithms are oriented towards the generation of prime implicates built up from a given subset of propositional variables (here, $ObsVar_t$); they do not require all the prime implicates to be generated, and are typically much more efficient than the naive filtering process that the definition of prime implicants/cates modulo a formula suggests (see [28] for more details).

At line (iii), we first check whether the selected decision covers or not some observations.

Algorithm 1 is correct w.r.t. its specification and has an anytime flavor.

Proposition 8

1. *At any step of the algorithm, σ is a sound partial policy;*
2. *If we let it run quietly, the algorithm stops* [9] *and returns a maximal sound policy.*

5 Related work and conclusion

Our main contribution is a purely propositional framework for qualitative one-stage decision making under uncertainty. In this framework, the existence of a solution policy is closely related to instances of quantified boolean formulas. We have analyzed it from the point of view of computational complexity, and pointed out an algorithm for computing solution policies (exactly if possible, approximately otherwise). It is also the first approach (as far as we know) which makes use of belief update in a decision-theoretic framework. As to planning using updates, the only approach we are aware of is the one by Brewka and Hertzberg [4] who encode the effects of actions by a disjunctive list of transitions $\langle precondition, postcondition \rangle$ and then minimize the set of violated transitions.

Now, the only approach we know that encodes planning problem into QBF is Rintanen's extension of the planning as satisfiability framework ([20]) to conditional planning [30]. His approach shares a lot with ours. Namely, his representation framework corresponds more or less to our STRIPS$^+$ assumption plus atomicity, and the update operator implicitly used for computing the consequences of actions corresponds to the MPMA with any of our choices 2. , 3. or 4. (which all coincide since one action only is performed at each step and effects are terms). However, he considers a multistage framework while ours is restricted to a single stage framework. At this point, it is worth noticing that our framework can be easily extended to fixed-horizon multistage decision making provided that *unobservability is assumed except possibly for the first stage*. With this assumption, we can represent a fixed-horizon planning problem by duplicating state variables and decision variables as many times as necessary as in SATPLAN; then, our policies map initial observations to unconditional sequences of actions.

[9] Within $\min(|AvDec(\mathcal{P})|, |PossObs(\mathcal{P})|)$ loops in the worst case.

As to the practical resolution of QBF, Cadoli et al. [6] propose algorithms for determining whether a given instance of QBF is true or false. This algorithm has been improved by Rintanen in [31]. Contrarily to us, they do not restrict their study to the second and third level of the polynomial hierarchy, and thus consider k-QBF and k-$\overline{\text{QBF}}$ for an arbitrary k. However, both their aim and their techniques differ a lot from ours, more so because their algorithms determine *whether a given QBF is true or not*[10] while ours aim at solving the *function problem* associated to QBF problems (namely, *computing policies*) which has more practical impact when dealing with decision making under uncertainty. The approach by Majercik and Littman [27, 25] is closer to ours since they encode a contingent probabilistic planning problem by a stochastic satisfiability problem (SSAT) which is close to a QBF problem (with "random" quantifiers instead of universal quantifiers). Technically, the resolution of the problem differs from ours (it does use prime implicants).

As to the logical representation of decision processes, the approach by [2] builds structured policies for fully observable, infinite-horizon, stationary Markov Decision Processes. Their algorithm, a structured version of modified policy iteration, incrementally splits application contexts so as to make the current policy converge towards an optimal one. The differences with our approach bear on the assumptions about the process (numerical data, infinite horizon, use of a discounting factor) and on the computational tools: their algorithm consists in iterately updating an "unsound", complete policy by splitting adequate application contexts, and does not make use of prime implicants. This kind of approach is extended to handling partial observability in [3].

We only know two approaches relating planning under incomplete information to abduction. In the first one [26], prime implicants are used to determine the conditions under which a plan always succeeds, may suceed, or always fails. This approach shares a lot with ours in what concerns the search for decisions covering an observation. The second one [32, 11] relates qualitative, *unobservable* decision making to abduction and propose a procedure to compute optimal decisions from stratified knowledge bases and goals. Unlike [26], they do not handle observations. Both approaches do not deal with the construction of policies.

The distinction between controllable and uncontrollable propositional variables goes back to [1] who also gives some abductive characterizations of suitable decisions given a knowledge base and a goal. However [1] does not consider observations and thus does not build conditional policies. The computational complexity of controllability and related notions is studied in [23]. [14] extend the classical CSP framework so as to solve problems of decision under uncertainty (and full observability). While their formalization has some similarities with ours, their algorithm requires strong restrictions on the syntax of the constraints involved (and is thus far less general than ours).

"Supermodels" [15] capture an alternative view of reactive decision under uncertainty: one looks for (unconditional) solutions which are guaranteed to be sufficiently close to a new solution if the data of the problem are slightly modified ("robustness"). Unsurprisingly, finding a supermodel has a lower complexity than finding a solution policy. This approach is based on a notion of distance which is completely absent from our framework (changes from one possible observation to another, or from one decision to another in a policy, are not required to be small); this makes their approach complementary to ours.

[10] Thus, in our framework, it can be used for checking the *existence* of a solution policy.

Acknowledgements

We thank Andreas Herzig and Omar Rifi for helpful discussions about update. The third author has been partly supported by a "Contrat d'objectifs de la Région Nord/Pas-de-Calais" and by the IUT de Lens.

References

[1] C. Boutilier. Toward a logic for qualitative decision theory. In *Proc. of KR'94*, pages 75–86, Bonn, 1994.

[2] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploting structure in policy construction. In *Proc. of IJCAI'95*, pages 1104–1111, Montreal, 1995.

[3] C. Boutilier and D. Poole. Computing optimal policies for partially observable decision processes using compact representations. In *Proc. of AAAI'96*, pages 1168–1175, Portland (OR), 1996.

[4] G. Brewka and J. Hertzberg. How to do things with worlds: on formalizing actions and plans. *Journal of Logic and Computation*, 3(5):517–532, 1993.

[5] T. Bylander. Complexity results for planning. In *Proc. of IJCAI'91*, pages 274–279, Sydney, 1991.

[6] M. Cadoli, A. Giovanardi, and M. Schaerf. An algorithm to evaluate quantified boolean formulae. In *Proc. of AAAI'98*, pages 262–267, 1998.

[7] M.-O. Cordier and P. Siegel. Prioritized transitions for updates. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty (Proc. of ECSQARU'95)*, Lectures Notes in Artificial Intelligence 946, pages 142–150, Fribourg, 1995. Springer-Verlag.

[8] A. Darwiche and J. Pearl. Symbolic causal networks. In *Proc. of AAAI'94*, pages 238–244, Seattle (WA), 1994.

[9] R. Dechter and I. Rish. Directional resolution: the Davis-Putnam procedure, revisited. In *Proc. of KR'94*, pages 134–145, Bonn, 1994.

[10] P. Doherty, W. Lukascewicz, and E. Madalinska-Bugaj. The PMA and relativizing change for action update. In *Proc. of KR'98*, pages 259–269, 1998.

[11] D. Dubois, D. Le Berre, H. Prade, and R. Sabbadin. Logical representation and computation of optimal decisions in a qualitative setting. In *Proc. of AAAI'98*, pages 588–593, Madison (WI), 1998.

[12] Th. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactual. *Artificial Intelligence*, 1992.

[13] Th. Eiter and G. Gottlob. The complexity of logic-based abduction. *Journal of the Association for Computing Machinery*, 42(1):3–42, 1995.

[14] H. Fargier, J. Lang, and Th. Schiex. Mixed constraint satisfaction: a framework for decision making under incomplete knowledge. In *Proc. of AAAI'96*, pages 175–180, Portland (OR), 1996.

[15] M.L. Ginsberg, A.J. Parkes, and A. Roy. Supermodels and robustness. In *Proc. of AAAI'98*, pages 334–339, Madison (WI), 1998.

[16] A. Herzig. The PMA revisited. In *Proc. of KR'96*, pages 40–50, 1996.

[17] A. Herzig and O. Rifi. Propositional belief base update and minimal change. *Artificial Intelligence*, 1999.

[18] R.A. Howard and J.E. Matheson. Influence diagrams. *The Principles and Applications of Decision Analysis*, 2:720–761, 1984.

[19] K. Inoue. Linear resolution in consequence–finding. *Artificial Intelligence*, 56(2–3):301–353, 1992.

[20] H. Kautz and B. Selman. Planning as satisfiability. In *Proc. of ECAI'92*, pages 359–363, 1992.

[21] J. Lang, P. Liberatore, and P. Marquis. The complexity of propositional independence. In preparation, 1999.

[22] J. Lang and P. Marquis. Complexity results for independence and definability. In *Proc. of KR'98*, pages 356–367, Trento, 1998.

[23] J. Lang and P. Marquis. Two forms of dependence in propositional logic: controllability and definability. In *Proc. of AAAI'98*, pages 268–273, Madison (WI), 1998.

[24] P. Liberatore. The complexity of belief update. In *Proc. of IJCAI'97*, pages 68–73, Nagoya, 1997.

[25] M. Littman. Initial experiments in stochastic satisfiability. In *Proc. of IJCAI'99*, pages 667–672, Stockholm, 1999.

[26] W. Lukascewicz and E Madalińska-Bugaj. Reasoning about plans. In *Proc. of IJCAI'97*, 1997.

[27] S. Majercik and M. Littman. Contingent planning under uncertainty via stochastic satisfiability. In *Proc. of AAAI'99*, pages 549–556, Orlando (FL), 1999.

[28] P. Marquis. *Consequence finding algorithms.* in The Handbook for Uncertain and Defeasible Reasoning, Volume 5 (Algorithms for Uncertain and Defeasible Reasoning). Kluwer Academic Publishers, 2000. To appear.

[29] Ch. H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[30] J. Rintanen. Constructing conditional plans by a theorem prover. *Journal of Artificial Intelligence Research (JAIR)*, 10:323–352, 1999.

[31] J. Rintanen. Improvements to the evaluation of Quantified Boolean Formulae. In *Proc. of IJCAI'99*, pages 1192–1197, Stockholm, 1999.

[32] R. Sabbadin. Decision as abduction. In *Proc. of ECAI'98*, pages 600–604, Brighton, 1998.

[33] L. Savage. *The foundations of statistics.* Wiley and Sons, 1954.

[34] L.J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[35] M. Winslett. Reasoning about action using a possible model approach. In *Proc. of AAAI'88*, pages 89–93, St Paul (MN), 1988.

Logical representation of preferences for group decision making

Céline Lafage **Jérôme Lang**

IRIT - Université Paul Sabatier

118, route de Narbonne
31062 Toulouse Cedex (France)
{lafage,lang}@irit.fr

Abstract

We propose two logical approaches for representing preferences in a group decision making context. The first one is based on weighted logics: each agent expresses her preferences by means of logical formulas weighted by importance degrees, and preferred decisions are those that satisfy best the collectivity of agents (according to some criteria). Aggregation is made in two steps: first the utility of each agent is computed from her individual preferences (independently from the other agents), and then individual utilities are aggregated into a collective utility function. We study the particular case corresponding to the pair of aggregation functions (sum,max) and show that some equity problems arise when the preferences expressed by the agents are not constrained enough. The second approach makes use of distances. We show that connections exist between both approaches.

1 Introduction

Group decision

Group decision making mainly consists of aggregating preferences of several agents into a collective preference ordering. The collective decision will then be a decision that is maximal with respect to the collective preference ordering (see, e.g., [19] or [18]). It is important to point out that the group eventually takes a unique, *collective decision* (contrariwise to game theory where each agent of a group acts separately). Group decision theory is composed of several classes of problems, well studied from a normative point of view in the field of economy. In a first class of problems, individual preferences of the agents are expressed quantitatively by means of *utility functions*; the collective preference ordering can then be represented by a collective utility function. In a second class of problems, individual preferences are expressed in an ordinal way by means of a *preference relation* over the set of possible solutions (or candidates); under this assumption, a *vote rule* maps a collection of individual preference relations into a single selected candidate and a *preference aggregation method* maps a collection of individual preference relations into a collective preference relation. Arrow's impossibility theorem is proved in the latter context: it tells that if an aggregation procedure satisfies a given series of rational postulates then the aggregation must be dictatorial, i.e., the collective preference relation must be equal to the preference relation of one of the agents.

Logical representation of preference

From the point of view of representation and computation, usual social choice methods (aggregation functions, voting rules etc.) can be *directly* applicable only when the respective numbers of agents (voters) and solutions (candidates) are reasonably low with respect to the available computational resources. This is always true when agents and candidates are individuals. Now, more generally, one may reasonably continue assuming that agents are still individuals, but we cannot any longer continue assuming that the number of candidates (or solutions) is not prohibitive. This happens in particular when the decision to take consists in giving a value to each of the variables of a set of n decision variables: here, the set of candidates is equal to the set of feasible assignments and thus grows exponentially with the number n of propositional variables. Therefore, it is obviously unreasonable to require from the agents the specification of an explicit utility function or preference ordering (under the form of a table or a list) on the set of all solutions. This argues for the need of a *compact* (or *succinct*) representation language for preferences. Furthermore, such a language for preference representation should be as *expressive* as possible, and as close as possible to the intuition (ideally, it should be easily translated from a specification in natural lan-

guage of the preferences of an agent). Lastly, it should be equipped with decision procedures, as efficient as possible, so as to enable the automation of the search for an optimal collective decision.

The KR community has contributed a lot to the study of such succint and expressive languages for representing *knowledge*[1]. Now, for the last few years, there has been several approaches dedicated to the representation of *preferences* of an agent, to be used in a decision making process. It is sometimes the case that a logic initially developed for reasoning about knowledge can be used efficiently for representing and handling preferences. This is for instance the case of default logic, which has been used for decision-theoretic issues [22][5], and of conditional logic [4]. More basically, the same remark applies to standard propositional logic, which can be seen as a (very) rough tool for representing preferences: the agent expresses her desires by specifying some logical formulas; feasible decisions are worlds, and optimal decisions consist of the worlds satisfying all formulas (worlds that violate any of the formulas being equally bad). This principle can be refined in the two following ways:

1. by associating to each formula a positive weight, namely, the penalty for not satisfying this formula; then, the penalty (or disutility) of a world is the sum of the weights of the formulas it violates [21, 12, 10]. If the penalties of the violated formulas are aggregated with *max* instead of sum, then we obtain a framework equivalent (up to a change of the valuation scale) to possibilistic logic [8]. We call *weighted logics* these logics built on this principle (propositional formulas + weights to be aggregated with a given function).
2. by grading the violation of a formula using a *distance* between worlds; given a formula φ representing a specific desire of the agent, and given a distance d, then the closer a world w to a model of φ, the better w (the optimal case being when w satisfies φ).

Outline of the paper

In this paper we develop the two above-mentioned approaches. For each of both, we start with the represention of individual preferences, and then we consider the case of a group of agents. We devote Section 2 to weighted logics for group decision making; in particular, we prove that given a set of reasonable postulates on how a utility function should be derived from a collection of individual preferences, the aggregation should be done in two successive steps (first at the individual level and then at the group level), thus leading to two (not necessarily identical) aggregation functions which respectively satisfy a number of specific properties. In Section 3, we study in further detail the case where the pair of aggregation functions is $\langle +, \max \rangle$, point on specific problems concerning equity between agents, and propose some solutions. In Section 4 we study distance-based logics for preference representation and we point out connections and intertranslations with weighted logics. Lastly, we position our work with respect to related work.

2 Weighted logics

Let $\mathcal{L}$ be a propositional language built from a finite set of variables and the usual connectives. Ω the set of interpretations (or worlds) associated with $\mathcal{L}$. Interpretations are denoted w, w_1 etc. formulas of $\mathcal{L}$ by Greek letters φ, ψ etc., and real numbers by x, y etc. $Mod(\varphi)$ is the set of all models of φ.

2.1 Individual preferences

Preferences are encoded in a compact and structured way under the form of elementary preference items; each of these items is a pair (formula, valuation) where the formula expresses a constraint the agent wishes to see satisfied and the valuation expresses the importance of this constraint. We choose $[0, +\infty]$ as valuation scale, once noted that this choice does not induce a loss of generality. Any closed infinite real interval could have been chosen instead, for instance $[0, 1]$ or $[-\infty, +\infty]$ since all infinite closed real intervals are isomorphic (the choice of $[0, +\infty]$ is merely motivated by considerations of intuition). The valuation $+\infty$ associated with a formula means that the constraint corresponding to this formula is absolute, while the valuation 0 expresses pure indifference towards the satisfaction of the constraint.

These weighted formulas defined above are concerned with the relative preference on some worlds over other ones, not with the feasibility of worlds. To express feasibility, we introduce *integrity constraints*. Integrity constraints should not be confused with hard preference constraints[2]. The former express some knowledge

[1] Here we reserve the term "knowledge" for its restrictive meaning "information about the real world", excluding thus preferences, goals and desires.

[2] In the literature of multi-agent system (see e.g. [23]) they are sometimes called respectively *hard* and *soft* constraints. We prefer to avoid this terminology which would be ambiguous here because agents may also express *subjective* hard constraints.

about the actual world and are thus *objective* (for instance, "it is not possible to be in vacation in Beijing and in San Francisco at the same time") while the latter express hard but *subjective* preferences (or goals) whose violation is unacceptable by the agent (for instance, "I do not want to kill"). A world violating an integrity constraint is an impossible world, while a world violating a hard subjective constraint is an unacceptable world.

Definition 1 (individual preference profile)
An individual preference profile *is a pair* $\langle P, K\rangle$ *where*

1. P *(*preference base*) is a multiset*[3] $\{(\varphi^j, \alpha^j)\}_j$ *with* $\varphi^j \in \mathcal{L}$, $\alpha^j \in [0, +\infty]$;
2. K *is a consistent set of formulas (*integrity constraints*).*

A world w is *possible* iff $w \models K$. Thus, the set of possible worlds is $Mod(K)$. In the rest of the paper we make the assumption that the decisions available to the collectivity of agents are precisely the possible worlds. This assumption is consistent with Boutilier's qualitative decision theory [4] when all atoms are controllable. Preference between worlds will be defined between possible worlds only. Namely, from P and K we build a *disutility* function, from $Mod(K)$ to $[0, +\infty]$, computed by aggregating the penalties of the constraints violated by a world:

Definition 2 (individual disutility)
The individual disutility function $disu_P$ *associated with the individual preference profile* $\langle K, P\rangle$ *is the mapping from* $Mod(K)$ *to* $[0, +\infty]$ *defined by*[4]

$$\forall w \in Mod(K) \quad disu_P(w) = *\{\alpha^i \mid w \models \neg\varphi^i\}$$

where $*$ *is an operation from* $[0, +\infty] \times [0, +\infty]$ *to* $[0, +\infty]$ *satisfying commutativity, non-decreasingness, associativity and* $\forall a,\ a * 0 = a$.

This disutility function, expressing the preferences of an agent under a numerical form, is thus computed by aggregating the valuations of all *violated* constraints. The justification of this computation scheme and of the four listed properties is discussed in Section 2.3. Noticeably, they imply $a * +\infty = +\infty$: indeed, since $*$ is non-decreasing, $a * 0 = a$ implies $a * b \geq \max(a, b)$ for all a, b, from which it follows that $+\infty$ is absorbent for $*$. If $disu(w) = +\infty$ then the possible world w is unacceptable by the agent.

[3] A *multiset* (or *bag*) is a set whose elements can appear several times.

[4] We should in principle mention K as well, i.e., we should note $disu_{K,P}$ but since when it is not ambiguous we drop K for making notations lighter.

Before justifying axiomatically the four listed properties, we comment them from an informal point of view. Commutativity is needed because the order in which the weighted formulas appear in P should be irrelevant. The need for non-decreasingness is obvious. 0 being neutral element of $*$ expresses that formulas with a null weight are irrelevant to the disutility function, and $+\infty$ being absorbent expresses that nothing can compensate the violation of an absolute constraint (whatever the other satisfied constraints, this world will have a maximal disutility). Associativity is needed by the requirement that the way the disutility of an agent is updated when taking a new constraint into account does not depend on where her initial disutility came from (this is the Decomposition postulate, cf. Section 2.3.). Noticeably, up to a change of scale from $[0, 1]$ to $[0, +\infty]$ and an order reversal, $*$ enjoys the same properties as *triangular norms*[5].

The most natural choices for $*$ are $+$ and *max*. The sum is an *utilitarianist* aggregation mode, allowing for compensations between violated formulas (as in penalty calculus [21]) while the maximum is a *pure egalitarianist* aggregation mode, where the disutility degree is computed relatively to the most important of the violated formulas (as in possibilistic logic [8])[6].

A disutility function induces in the usual way a weak total order, i.e., a transitive, reflexive and total relation, on the set of possibles worlds: a possible world w_1 is preferred to the possible world w_2 (for a given agent) iff $disu_P(w_1) \leq disu_P(w_2)$.

2.2 Collective preference aggregation

We now consider a group of n agents, whose individual preferences are represented by means of n individual preference bases.

Definition 3
A multi-agent preference base (MAPB) is a collection $(P_1, ..., P_n)$ *of individual preference bases where* $P_i = \{(\varphi_i^j, \alpha_i^j)\}_j$. *A multi-agent preference profile consists of a MAPB and a set of integrity constraints* K.

[5] A triangular norm is a mapping $T : [0, 1]\times[0, 1] \to [0, 1]$ satisfying commutativity, monotonicity, associativity and with 1 as neutral element. A common bijective transformation f from $[0, +\infty]$ to $[0, 1]$ is $f(x) = e^{-x}$; note that the ordering is then reversed ($a \leq b$ iff $f(a) \geq f(b)$, $f(0) = 1$ and $f(+\infty) = 0$).

[6] Note that, except in the case of pure utilitarianism, it is important that P should be a multiset and not a set (for instance, if the aggregation function is $+$, the preference base $\{(a\ 1), (a\ 1)\}$ is not equivalent to $\{a\ 1)\}$ but to $\{a\ 2\}$. Further comments can be found on this and more generally on dependence between formulas in this context can be found in [10].

The collective disutility function is computed in two steps: first, the individual disutility functions are computed independently (as above), and then they are aggregated into a collective disutility function that best conveys the preferences of the group (see figure 2.2).

Definition 4 (collective disutility)
The collective disutility function of a MAPB P is defined as:

$\forall w \models K \quad disu_P(w) = \Diamond\{disu_i(w) \mid i = 1, ..., n\}$

where

- *$disu_i$ is the disutility function of agent i computed from P_i.*
- *$\Diamond$ is a function from $[0, +\infty]^n$ to $[0, +\infty]$ satisfying non-decreasingness in each of its arguments and commutativity.*

ω^* is optimal *w.r.t. P iff $disu_P(\omega^*) = \min_{w \models K} disu_P(w)$.*

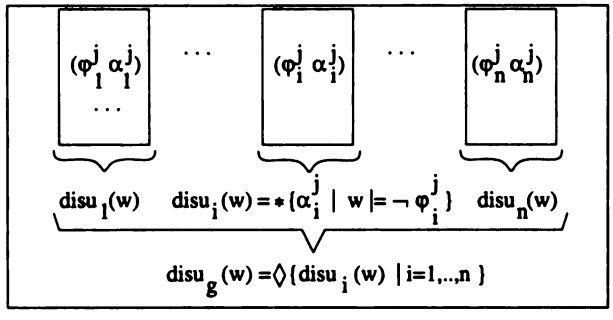


Figure 1: Preferences bases

The only two properties required for $\Diamond$ are non-decreasingness in each of its arguments and commutativity. Commutativity guarantees *anonymity* between agents because a permutation of preference bases of agents do not change global disutility. Non-decreasingness is required for obvious reasons. Among the possible $\Diamond$ operators, we again find *max* and $+$ that still correspond to egalitarianism and utilitarianism (in a more usual meaning than for $*$ – because these notions primarily come from group decision theory). *min* would be theoretically acceptable, but it would not be intuitively satisfactory since it compares decisions with respect to the *most* satisfied agent. An interesting class of acceptable operators consists of the "Ordered Weighted Average" operators (OWA) [26]. An OWA is an application $F : \mathbb{R}^n \rightarrow \mathbb{R}$ associated with a vector of weights $\vec{x} = [x_1, x_2, ..., x_n]$ with $x_i \in [0, 1]$ and $\sum_{i=1}^{n} x_i = 1$, such that $F(a_1, a_2, ..., a_n) = \sum_{i=1}^{n} x_i b_i$ where $(b_1, ..., b_n)$ is a reordering of $(a_1, ..., a_n)$ such that $b_1 \geq b_2 ... \geq b_n$. Distinguished OWAs are *min*, *max* and the mean, associated respectively to the vectors $(0, ..., 0, 1)$, $(1, 0, ..., 0)$ and $(\frac{1}{n}, ..., \frac{1}{n})$. More refined OWAs express a compromise between *min* and *max* to a given degree. Note that OWAs are generally not associative.

Therefore, a weighted logic will be associated with a pair of operators $\langle *, \Diamond \rangle$. Choosing $*$ is independent from choosing $\Diamond$. Noticeably, if $*$ and $\Diamond$ coincide then preferences of agents 1, ..., n are gathered as if there were only one agent, which comes down to a mono-agent weighted logic (thus, $\langle +, + \rangle$ corresponds to penalty logic). Now, when $*$ and $\Diamond$ do not coincide, some associations are more satisfactory than others from the point of view of intuition. In particular, egalitarianism is intuitively more relevant between agents than between the constraints expressed by a single agent, which makes the choice $\langle +, \max \rangle$ a priori more natural than the choice $\langle \max, + \rangle$. The former ($\langle +, \max \rangle$) will be studied in more detail in Section 3.

2.3 Axiomatic justification

Let $P = \{P_1, ..., P_n\}$ be a MAPB and K a set of integrity constraints. We introduce the following notations:

- $Add(P, i, (\varphi\ \alpha))$ is the MAPB obtained from P by replacing P_i by $P_i \cup \{(\varphi\ \alpha)\}$;
- $Delete(P, i, (\varphi\ \alpha))$ is obtained from P by deleting $(\varphi\ \alpha)$ from P_i (and if $(\varphi\ \alpha)$ $/in P$ then $Delete(P, i, (\varphi\ \alpha)) = P$).
- $Replace(P, i, (\varphi\ \alpha), (\varphi'\ \alpha'))$ is obtained from P by substituting $(\varphi\ \alpha)$ by $(\varphi'\ \alpha')$ in P_i;
- $Neutralize(P, i) = \{P_1, ..., P_{i-1}, \emptyset, P_{i+1}, ..., P_n\}$ is obtained from P by forgetting the preferences of agent i.

Now, we define the following postulates. The first one is distinguished from the following ones because it pertains to the integrity constraints.

Insensitivity towards integrity constraints:
Let K_1, K_2 two sets of integrity constraints; then for any $w \models K_1 \wedge K_2$, we have
$disu_{(P,K_1)}(w) = disu_{(P,K_2)}(w)$. (IC)

Thus, when (IC) is satisfied we can merely note $disu_P(w)$ instead of $disu_{(P,\emptyset)}(w)$. We implicitely assume (IC) when writing the following axioms, thus enabling the fixing of K and using the above simplified notation.

Anonymity:
Let $P_\sigma = (P_{\sigma(1)}, ..., P_{\sigma(n)})$ where σ is a permutation of $\{1, ..., n\}$; then
$\forall w \models K \quad disu_P(w) = disu_{P_\sigma}(w)$. (A)

Neutrality:
Let $w, w' \in Mod(K)$ be two possible worlds such that for any weighted formula $(\varphi_i^j \ \alpha_i^j)$ appearing in one of the P_i's, the following equivalence holds: $(w \models \varphi_i^j) \Leftrightarrow (w' \models \varphi_i^j)$. Then $disu_P(w) = disu_P(w')$. (N)

Irrelevance of the syntax:
Let $P' = Replace(P, i, (\varphi_i^j \ \alpha_i^j), (\varphi'^j_i \ \alpha_i^j))$, with $\varphi_i^j \wedge K \equiv \varphi'^j_i \wedge K$; then
$\forall w \models K$, $disu_P(w) = disu_{P'}(w)$. (IS)

Insensitivity to satisfied formulas:
Let $P' = Replace(P, i, (\varphi \ \alpha), (\varphi \ \beta))$; then
$\forall w \models K \wedge \varphi$, $disu_{P'}(w) = disu_P(w)$. (ISF)

Monotonicity:
Let $P' = Replace(P, i, (\varphi \ \alpha), (\varphi \ \beta))$ with $\beta > \alpha$;
$\forall w \models K \wedge \neg\varphi$, $disu_{P'}(w) \geq disu_P(w)$. (MON)

Strict monotonicity:
as above, replacing $disu_{P'}(w) \geq disu_P(w)$ by $disu_{P'}(w) > disu_P(w)$. (MONS)

Insensitivity to constraint ordering:
If P' is obtained from P by exchanging the ordering of the constraints $(\varphi_i^j \ \alpha_i^j)$ in one of the P_i then $disu_P = disu_{P'}$. (ICO)

Priority to hard constraints:
If one of the P_i contains $(\varphi_i^j \ +\infty)$ then
$\forall w \models K \wedge \neg\varphi_i^j$ we have $disu_P(w) = +\infty$. (H)

Insensitivity to 0:
Let $P' = Add(P, i, (\varphi \ 0))$, then
$disu_P = disu_{P'}$. (I0)

Decomposition:
Let $P' = Add(P, i, (\varphi \ \alpha))$ and $Q' = Add(Q, i, (\varphi \ \alpha))$, if $\forall j \neq i \quad P_j = \emptyset$ *and* $Q_j = \emptyset$ then
$(disu_P = disu_Q) \Rightarrow (disu_{P'} = disu_{Q'})$. (DEC)

Transfer:
Let $P' = Add(Delete(P, i, (\varphi_i^j \ \alpha_i^j)), k, (\varphi_i^j \ \alpha_i^j))$, i.e. P' is obtained from P by transferring $(\varphi_i^j \ \alpha_i^j) \in P_i$ from P_i to P_k, then
$disu_P = disu_{P'}$. (TR)

Separation:
If $disu_{Neutralize(P,i)} = disu_{Neutralize(P',i)}$ and $P_i = P'_i$ then $disu_P = disu_{P'}$. (SEP)

Initialization:
If $P = \{\emptyset, ..., \emptyset\}$ then $\forall w \models K$, $disu_P(w) = 0$. (INIT)

Insensitivity towards integrity constraints mean that the way the disutility of a possible world is computed does not depend on where the possible worlds are. This postulate is reminiscent of the independence from irrelevant alternatives assumption is social theory [19].

Anonymity and neutrality are two central properties in social choice theory [19]. *Anonymity* ensures equality between agents, while *neutrality* ensures initial equality between possible worlds[7].

The following group of postulates deal specifically with the influence of valuations on the computation of the disutility function. *Monotonicity* is very natural and does not need any comments. *Strict monotonicity* is much more demanding since it rules out some natural $\Diamond = max$. *Priority to hard constraints* is rather natural and can be ignored simply by forbidding agents to specify constraints with infinite penalties. *Insensitivity to 0* expresses the neutrality of the valuation 0. Of course, this neutrality is a consequence of the assumption about negative preferences expressed by (ISF) (see next paragraph). Whether 0 should be neutral or not has been discussed a lot in social decision theory. For practical applications in KR it does not seem to be harmful.

Several postulates deal more specifically with representational issues.

Insensitivity to constraint ordering tells that the ordering in which a given agent has expressed her constraints does not matter. *Irrelevance of the syntax* tells that the logical content only of formulas is relevant when computing disutilities. *Insensitivity to satisfied formulas* is a strong assumption. It says that the disutility degree of a world is computed from the constraints whose formulas are violated by this world, only. Or, equivalently, that given a world w, adding to the preference base of an agent any constraint satisfied by w does not change her disutility. This central assumption means that *preferences are represented negatively*, i.e., by a set of constraints (whose violation in-

[7] An aggregation rule that would not be neutral is the following one: let us consider a collection of agents in a bus expressing their preferences about opening or closing the window; an aggregation rule such as: *if* $w \models \neg windowopen$ *then* $disu_P(w) = \max_i disu_i(w)$ *else* $disu_P(w) = \frac{\sum_i disu_i(w)}{n}$ is anonymous but not neutral since it favours worlds in which the window is closed; an aggregation rule such as $\forall w \models K, disu_P(w) = \sum_i \lambda_i disu_i(w)$ *where* λ_i *increases with the age of agent* i is neutral but not anonymous.

duces a loss of utility) — in particular, an agent cannot be strictly more satisfied than when she does not specify any constraint. Positively represented preferences would consist in *goals*, i.e., formulas whose satisfaction *increases the utility*. This assumption of negative preferencesd is present in all penalty-based approaches [21] [10] [12] and is often well-suited to practical applications. Van der Torre and Weydert [25] formalize this distinction between constraints and goals using the notion of *polarity* and consider as well preference items being intermediate between constraints and goals. An extension of our axiomatic system to *positive preferences (goals) only* can be inferred easily by replacing insensitivity to satisfied formulas by insensitivity to violated formulas, and replacing disutilities by (positive) utilities. An extension to preferences with mixed or intermediate polarities is left to further research. *Initialization* is concerned both with the neutral role of 0 and the negative representation of preference, since, together with monotonicity and (ISF), it entails that 0 is the minimal disutility, and that this minimal idusility is reached in particular when agents do not express any constraints.

Decomposition and *Separation* deal with the preservation of the indifference (i.e., the utility equivalence) of two MAPBs when, respectively, a new constraint is added to both of them (to the same agent and with the same weight in both) and when a new preference base is added to both of them.

Lastly, *Transfer* is a very strong assumption (and often undesirable) since, by allowing to tranfer any constraint from one preference base into any other one, it implies that agents are not considered as individuals any longer and that the MAPB becomes equivalent to the single-agent preference base obtained by gathering the constraints expressed by the agents.

Proposition 1
A function $\Phi : \langle P, K\rangle \mapsto disu_{\langle P,K\rangle}$ *mapping each pair* $\langle P, K\rangle$ *into a disutility function* $disu_{\langle P,K\rangle}$ *on* $Mod(K)$ *satisfies (A), (N), (MON), (IS), (ISF), (ICO), (I0), (DEC), (SEP), (INIT) and (H) iff there exists a function* $* : [0,+\infty] \times [0,+\infty] \to [0,+\infty]$ *commutative, associative, non-decreasing, such that* $\alpha * 0 = \alpha$ *and* $\alpha * +\infty = +\infty$, *and a function* $\Diamond : [0,+\infty]^n \to [0,+\infty]$, *commutative and non-decreasing in each of its arguments, such that* $disu_P(w) = \Diamond\{disu_1(w), disu_2(w), ..., disu_n(w)\}$ *where* $disu_i(w) = *\{\alpha_i^j \mid w \models \neg\varphi_i^j\}$.

Proposition 1bis: *A function* $\Phi : \langle P, K\rangle \mapsto disu_{\langle P,K\rangle}$ *satisfies (A), (N), (MONS), (IS), (ISF), (ICO), (I0), (DEC), (SEP), (INIT), (H) iff there exists two functions* $*$ *and* $\Diamond$ *as previously and also verifying:* $*$ *and* $\Diamond$ *satisfy also strict monotonicity.*

Proposition 2 *A function* $\Phi : \langle P, K\rangle \mapsto disu_{\langle P,K\rangle}$ *satisfies (A), (N), (MON), (IS), (ISF), (ICO), (I0), (DEC), (SEP), (INIT), (H) and transfer (TR) iff there exists a function* $* : [0,+\infty]^2 \to [0,+\infty]$ *commutative, associative, non-decreasing, such that* $\alpha * 0 = \alpha$ *and* $\alpha * +\infty = +\infty$, *and such that:*

$$disu_P(w) = *\{\alpha_i^j \mid w \models \neg\varphi_i^j,\ i = 1..n\}.$$

Proofs are in [15]. Note that strict monotonicity (in Proposition 2) excludes the idempotent operators *min* and *max*. Proposition 3 characterizes weighted logics for which $*$ and $\Diamond$ coincide.

2.4 Comments

2.4.1 Weighted logics and Arrow's impossibility theorem

It is well-known that collective choice is a difficult task, especially because numerous impossibility theorems exist; each of these consider a set of reasonable postulates and show its inconsistency, which means that there cannot be any procedure for collective choice satisfying these postulates. The most well-known of these theorems is Arrow's [1], which (roughly) states the following: given a society of n agents and a set of $p \geq 3$ candidates, there is no mapping aggregating *any* collection of individual preference relations consisting of *total orderings over candidates* into a collective preference consisting of a total preordering (indifference is allowed) over candidates and satisfying the three following conditions: (i) *unanimity*: if every agent prefers a candidate a to a candidate b then a is collectively preferred to b; (ii) *independence of irrelevant alternatives (IIA)*: for any two candidates a and b, the collective preference between a and b depends only upon individual preferences over that pair; (iii) *non-dictatorship*: there is no individual such that for any two candidates a and b, a is collectively preferred to b whenever this individual prefers a to b.

There are three main ways of escaping Arrow's theorem. The first one consists of relaxing one of the conditions (most of the times, IIA). The second one consists in restricting the domain of possible collections of individual preference relations (or, equivalently, in dropping the requirement that the mapping should be defined for any collection of individual preferences). The third one consists in changing the nature of the input, *i.e.*, dropping the assumption that individual preference should be total orderings over candidates. Clearly enough, weighted logics, by enabling agents expressing preferences *quantitatively* and not under the form of purely ordinal preferences, use the third of

the above ways to escape Arrow's theorem. As explained in [13], things are similar for belief merging [14]. The main consequence of this choice is that *preferences should be expressed quantitatively and not in a purely ordinal way.* One the one hand, this has the further advantage to enable agents to express intensities of preferences. On the other hand, this leads to another well-known problem in social choice theory, namely, *interpersonal comparison of utilities.* This issue is discussed in the next paragraph.

2.4.2 Utility assessment and interpersonal comparison of utilities

Encoding preferences by means of formulas weighted by a numerical disutility weight raises several comments (see the Example in Section 3.1 for an illustration). In particular, one may wonder about *how weights are assessed.* A first (but unsufficient) element of answer is the following: we assume that a preliminary (upstream) task of *preference elicitation* consisting in asking each agent questions so as to find how prioritary each other her desires are, and also so as to assess independence relations between the satisfaction of her different desires (see Section 3.1). Now, there is a more tricky question, namely, the problem of *interpersonal comparison of utilities.* This issue is central to group decision theory: as explained for instance in [18], since the utility function representing the true preferences of an (utilitarianist) agent is not unique[8], it is *a priori* meaningless comparing utilities of different agents; however, quoting [18], *in certain contexts, (...) it is reasonable to assume that a commensurating unit and a common origin of measurement exist.* This is indeed the case for many practical problems in restricted domains.

2.5 Computational complexity

Although this paper does not focus on how to compute effectively optimal group decisions, we evoke here briefly some questions regarding computational complexity. The necessary background (omitted for considerations of space) can be found in Papadimitriou's textbook [20], especially in Chapter 17.

When $* = \diamond = \max$ (which makes the framework equivalent to possibilistic logic whose complexity issues are investigated in [16]), finding an optimal assignment for a given multi-agent preference profile can be done using $\mathcal{O}(\log n)$ queries to a satisfiability or unsatisfiability problem. This result is due to these two facts: (i) for any α, the problem of determining whether there exists a possible world w such that $disu_P(w) \leq \alpha$ is coNP-complete; (ii) the use of *max* as aggregation function does not create any new valuations, i.e., $disu_P(w) = x$ iff there is a $(\varphi\ x)$ in one of the P_i's. Therefore, when $* = \diamond = \max$, finding an optimal assignment is in $\mathsf{FP}^{\mathsf{NP}[\mathcal{O}(\log n)]}$. This result holds as well for the general case when penalties appearing in the preference bases are all identical (for instance, unitary), and for the same reasons (i) and (ii) as above. In the general case, however, this result does not hold and what we have is that finding an optimal assignment is in $\mathsf{FP}^{\mathsf{NP}[\mathcal{O}(n)]}$. The complexity of more general problems concerning weighted logics is investigated in [17].

3 A case study: $\langle +, \max \rangle$

We now choose that preference aggregation between constraints of a given individual is utilitarianist and that aggregation of preferences coming from different agents is utilitarianist (cf. end of Section 2.3).

3.1 An example

Example 1: Three friends (named 1,2,3) are organizing their next vacation. If they decide to spend their vacation together then they have to choose when to go (in summertime S or in wintertime W) and where (in a hot place H or not, in the mountains M or not). 1 likes very much the idea of common vacation, and then prefers to go in summertime and prefers hot places. 2 would really like to go in the mountains and preferably in wintertime. 3 desires very much having vacation and slightly prefers going to the sea in summer. The company permits at most one vacation period (therefore it is not possible to go in vacation both in wintertime and in summertime); there is no hot place in wintertime; moreover, it is never hot in the mountains.

These specifications are encoded by
$K = \{V \rightarrow (S \leftrightarrow \neg W), W \vee M \rightarrow \neg H\},$
$P_1 = \{(V\ 60), (V \rightarrow S\ 20), (V \rightarrow H\ 20)\},$
$P_2 = \{V\ 50), (V \rightarrow M \wedge W\ 25), (V \rightarrow M\ 25)\},$
$P_3 = \{(V\ 80), (\neg M\ 10), (S\ 10)\}.$

This encoding raises several comments. In particular, as evoked in Section 2.4.2, one may wonder about *how this weights are assessed. Why these ones and not other ones?* Let us consider the preferences expressed by agent 2. The weights imply (and should come from) that 2 is indifferent between having vacation in a disliked place at a disliked period and not having vacation, and that the difference of utility be-

[8] Namely, if u is a utility function representing the preferences of a given agent, then this is also the case of any positive linear combination $au + b$ $(a > 0)$.

tween "mountain in winter" and "moutain in summer" is the same as between "moutain in summer" and "not in mountain". As to agent 3, the choice of translating that he slightly prefers going to the sea in summer by $\{(\neg M\ 10), (S\ 10)\}$ expresses independence between these two desires, contrariwise to $\{\neg M \wedge S\ 10)\}$.

Let $w_1 = (V, S, M, \neg H)$, $w_2 = (V, S, \neg M, H)$ and $w_3 = (V, W, M, \neg H)$. We have $disu_1(w_1) = 20$, $disu_2(w_1) = 25$, $disu_3(w_1) = 10$, thus $disu(w) = \max(20, 25, 10) = 25$; $disu(w_2) = \max(0, 50, 0) = 50$ and $disu(w_3) = \max(40, 0, 20) = 40$. It can be checked that w_1 is optimal (for the pair of operators $\langle +, \max \rangle$). If we had chosen $\langle +, + \rangle$ (resp. $\langle \max, \max \rangle$) then the optimal world would have been w_2 (resp. w_3). Note that the three agents are indifferent between any two worlds satisfying both $\neg V$ (which reflects that the variables S,W,H,M are irrelevant when there is no vacation). For any such w we have $disu_1(w) = 60$, $disu_2(w) = 50$, $disu_3(w) = 80$.

3.2 Fairness issues

Expressing preferences in logic in a group decision context raises some issues dealing with fairness. We discuss and illustrate these problems in the case of the weighted logic associated to $\langle +, \max \rangle$ but we believe that similar problems would hold not only with other aggregation operators but more generally with any logical preference representation language.

3.2.1 Scale sensitivity

Let $P_1 = \{(A\ 60), (B\ 40)\}$ and $P_2 = \{(\neg A\ 6), (\neg B\ 4)\}$. Obviously, specifying larger weights favours agent 1 since the optimal world is (A, B). This is easily solved by requiring the sum of the penalties of each agent to be constant (this was the case in the previous example). Let us now focus on something less obvious.

3.2.2 Sensitivity to constrainedness

Example 2: Let
$P_1 = \{(A \wedge B \wedge C \wedge \neg D\ 100)\}$,
$P_2 = \{(\neg A\ \wedge \neg B\ 50), (\neg C\ 25), (\neg D\ 25)\}$ *and*
$P_3 = \{(\neg A \vee \neg B\ 50), (\neg C \vee \neg D\ 50)\}$.

The disutility function of each of these 3 agents is shown on figure 3.2.3. We notice that the only world that does not induce a maximal disutility to agent 1 is $w_2 = (A, B, C, \neg D)$ and thus it is the only $\langle +, \max \rangle$-optimal world (since the global disutility degree of a world is the maximum of the disutility degrees of each agent for this world). Consequently, agent 1 imposes her preferences to agents 2 and 3. What favours her is not the sum of the weights she assessed but their repartition: first, she chose to put the whole weight on one formula only, so that every world violating this formula has a maximum disutility degree; then, more importantly, this formula has only one model, which means that the whole weight was used to express that *one world is fully acceptable, and all other worlds are unacceptable.* In contrast, agents 2 and 3 have more balanced disutility distributions. For agent 2, one world satifies all constraints, two worlds are rather acceptable and only three have maximum disutility. Agent 3 is even more helpful: nine worlds fully satisfy her, six are reasonably good to her (disutility 50) and only one has maximum disutility.

The problem is rather complex and concerns both (a) the *repartition* of weights on formulas and (b) the *constrainedness* of the formulas — namely, specifying formulas with a small set of models such as large conjunctions (c.f. agent 1) gives an advantage over specifying formulas with a larger set of models. This encourages agents to be unsincere, i.e. to hide their true preferences: thus, an agent would tend to express $\{(X \wedge Y\ 50)\}$ instead of $(X\ 25), (Y\ 25)\}$ even if her desires for X and Y are independent. For instance, it is not unreasonable to assume that agent 1 in unsincere when she expresses P_1: it may actually be the case that in reality, her preferences for A, B, C and $\neg D$ are independent, which corresponds $P_1' = \{(A\ 25), (B\ 25), (C\ 25), (\neg D\ 25)\}$. Clearly, P_1' does not favour agent 1 as much as P_1 does.

3.2.3 Towards solutions

The abovementioned fairness problems are the consequence of the *freedom* given to agents to specify preference bases as they like. To remedy them, two general classes of solutions can be considered:

(1) *Imposing a format for preference specification.*

Under this assumption, agents are required to respect some syntactical constraints (i) on how the weights are distributed and, especially, (ii) on the formulas allowed (for instance, we may allow for literals only, or disjonctions of literals only, or allowing for a few conjunctions with a bounded number of literals, etc.). This suffers from the severe drawback of loosing expressivity, even if this solution can be satisfactory in many practical situations. For instance, suppose we forbid conjunctions then some desires cannot be expressed, especially in case of dependencies: how would an agent express without conjunctions that she wants to go skiing if the language is composed only of the variables M(ountain) and W(inter)? Clearly, expressing M and W in two separate constraints means something different.

(2) *Postprocessing the disutility distribution.*

Under this assumption, agents are completely free to specify any kind of constraints but once the disutility function computed, it is *postprocessed* so as to homogeneize the disutility function between agents. For instance, normalizing the sum of the penalties distributed is a straightforward postprocessing task (which is equivalent to imposing a fixed penalty amount); as shown above, this normalization is not sufficient. Let us consider Example 2 again. What is needed is a reequilibration of the disutility distribution. To measure how demanding an agent is (i.e., how constrained her disutility function is), a possible approach consists in *measuring the repartition of the distility with an entropy index* and then *transform the distributions so as to make their entropy close — ideally, equal.* This calls for some technical comments. First, entropy is usually computed for a probability distribution, not for a (dis)utility distribution. In order to make things homogeneous, these disutilities have to be redefined so as to be technically equivalent to probability distributions. Then, knowing that the extreme cases are (a) equidisutility and (b) one fully acceptable world only and all other ones fully unacceptable, it is obvious that we first have to transform disutilities into (normalized) utilities. This transformation is the following: let $maxd_i$ be the sum of all penalties given by agent i, and $maxd = \max_{i=1..n} maxd_i$. Clearly enough, no world can have a collective disutility larger then $maxd$ and for any agent i, no world can have a disutility larger than $maxd_i$. Then, for any agent i and any possible world w, let

$$u_i(w) = \frac{maxd_i - disu_i(w)}{\sum_{w' \models K} (maxd_i - disu_i(w'))}$$

Clearly, we have $u_i(w) \in [0,1]$ and $\sum_{w \models K} u(w) = 1$, which means that u is, *technically speaking*, a probability distribution[9] . We are now in position to compute its entropy index:

$$H(u_i) = - \sum_{w \models K u_i(w) \neq 0} u_i(w) \ln u_i(w)$$

Note that H increases with the "utility dispersion"; in particular, the entropy is minimal (= 0) in the extreme case when one world is acceptable and all the other ones fully inacceptable (cf. agent 1 in Example 2) and maximal (= $\ln |Mod(K)|$) when all worlds are equally preferred. In example 2, we have $H(u_1) = 0, H(u_2) = 2.44, H(u_3) = 2.66$. The maximal disutility index possible would be $\ln 16 = 2.77$.

Now, the postprocessing task consist of renormalizing the disutility $disu_i$ (or equivalently the utility u_i) of each agent so as to make the entropy indices tending to be equal – ideally, all equal to a fixed quantity x such that $0 < x < \ln |Mod(K)|$. For (demanding) agents with low entropy, the renormalization will decrease the disutility of very disliked worlds. Symetrically, for (non-demanding) agents whose entropy is too high, the disutility of disliked worlds is decreased. There are several possible choices for doing such a transformation (the simplest one consisting of homothetic transformations) and we do not go give them in detail. It is nevertheless important to emphasize that the candidate transformations should not be order-preserving and 0-preserving (worlds with an initial disutility degree of zero should keep this null utility). It may be the case that for some very little demanding agents, their (dis)utility cannot be modified so as to make the entropy index qual to the fixed quantity. This is for instance the case for an agent i such that u_i is constant ("equi-utility"). But this is not a problem, since there is no reason to create arbitrary preferences between worlds for an agent who has none!

Another (drastic but efficient) trick for renormalizing disutility functions consists of fixing a minimum number of worlds with a null disutility, and if an agent does not respect this condition, then one decreases down to 0 the disutility degrees of his preferred worlds; consequently, being too much demanding, for instance all the weight on a single world such, becomes a risky behaviour (this seems rather fair, since the agent is not very cooperative).

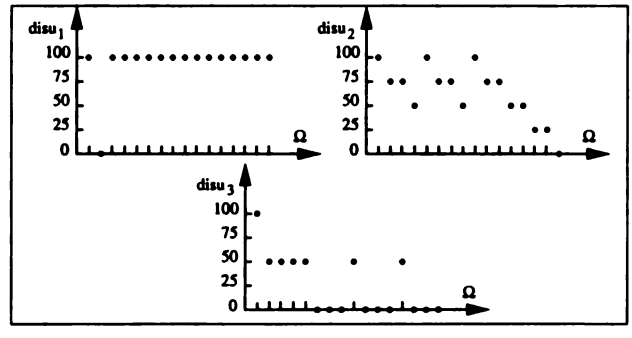


Figure 2: Example 2

4 Distance-based logics

The notion of distance plays an important role in knowledge representation, especially for belief revision, belief update, merging, and similarity-based reasoning

[9] We insist on the purely *technical* aspect of this transformation. From a semantical perspective, probabilities and utilities are completely different: probabilities represent uncertainty while utility represent preferences.

(see [3] for a discussion). We show here that it also plays a role for preference representation and group decision making. This has been already (implicitly) remarked in [14]: although their primary motivation is the merging of *beliefs*, some of their examples show that their framework can also be used for aggregating preferences in a group decision context. We now explore this issue further and show a formal parallel between disutilities and distances; the intuition being that if a formula G represents some goal an agent wants to be satisfied, then the closer to G the world w is, the more preferred it is w.r.t. the satisfaction of G.

Definition 5 *A* distance *is a mapping* $d: \Omega \times \Omega \rightarrow IN$ *such that*

(i) $\forall w, w' \in \Omega, d(w,w') = 0 \Leftrightarrow w = w'$;
(ii) $\forall w, w' \in \Omega, d(w,w') = d(w',w)$;
(iii) $\forall w, w', w'' \in \Omega,\ d(w,w'') \leq d(w,w') + d(w',w'')$.

A weak distance *is a mapping satisfying (ii), (iii) and (i')* $\forall w \in \Omega, d(w,w) = 0$.
Let φ *be a formula and* w, w' *two worlds; we define* $d(w,\varphi) = \min_{w' \in Mod(\varphi)} d(w,w')$ *and* $d(\varphi,\psi) = \min_{w \models \varphi, w' \models \psi} d(w,w')$.

A commonly used distance is the Hamming distance: $d_H(w,w')$ is defined as the number of propositional symbols that are assigned different values in the two worlds w, w'. It can be generalized assigning weights to each propositional symbol, the resulting distance being the weighted Hamming distance: $\forall w, w' \in \Omega \quad d_{WH}(w,w') = \sum_{v_i \in \mathcal{V}, w(v_i) \neq w(v_i)} p_i$.
If $\forall v_i \in \mathcal{V} \quad p_i \neq 0$ then d_{WH} is a distance, while if we allow the p_i to be null, i.e. the value of the corresponding propositional symbol has no importance, then d_{WH} is only a weak distance.
From now on we take by default $d = d_H$.

4.1 From distances to weighted formulas and vice-versa

A distance-based group decision making problem then consists of the data of a distance (assumed identical for all agents – but this constraint could be relaxed) and, for each agent i, a set of goals G_i^j. The disutility of agent i is then computed as $disu_i(w) = *\{d(w, G_i^j) | j\}$ where $*$ is an operator as in Section 2, and inter-agent aggregation is exactly as in Section 2. What remains for us is to exhibit a translation between the specification of a set of weighted formulas and the specification of a set of goals together with a distance. First of all, we need the following definition of a *disc around a formula*, obtained by gathering all worlds that are at most at a given distance from the formula. Formally: for $i \in IN$, $D(\varphi, i)$ represents a formula such that $Mod(D(\varphi,i)) = \{w \mid d(w,\varphi) \leq i\}$ (as an example, see figure 4.1. Clearly, we have $D(\varphi,0) = \varphi \models D(\varphi,1) \models \ldots \models D(\varphi,m) = \top$, where m is the number of propositional variables in the language.

Clearly, we have $D(\varphi_1 \vee \ldots \vee \varphi_p, k) = D(\varphi_1, k) \vee \ldots \vee D(\varphi_n, k)$. Moreover, if $\gamma = l_1 \wedge \ldots \wedge l_q$ is a consistent conjunction of literals, then it can easily be shown that $\forall k \leq q \quad D(G,k) = \bigvee_{I \subseteq \{1,\ldots,q\}, |I| = q-k} \bigwedge_{i \in I} l_i$

In particular, we have
$D(\gamma, 0) = \gamma, \quad D(\gamma, 1) = \vee(\bigwedge_{i \neq j} l_i), \quad D(\gamma, q) = \top$.

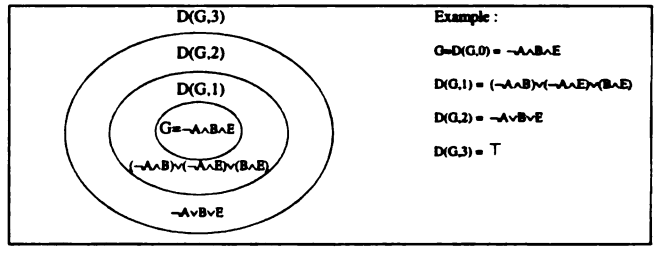


Figure 3: Goals and distance discs

These relations show that computing the disc around a DNF formula is easy:

Proposition 3
For $\varphi = \gamma_1 \vee \ldots \vee \gamma_p$ *in DNF,*
$D(\varphi,k) = \bigvee_{\substack{i=1\ldots p \\ |I| = q-k}} \bigvee_{\substack{L \subseteq Lit(\gamma_i) \\ |L| = |\gamma_i| - k}} \bigwedge_{l \in L} l$

Therefore, when φ is under DNF, (a) $D(\varphi,k)$ can be computed in time $\mathcal{O}(n^k)$ and (b) $D(\varphi,k) = \top$ as soon as k reaches the number of literals of the longest of the γ_i's, i.e. as soon as $k \geq \max_{\gamma_i} |\gamma_i|$.

We are now able to establish the formal connection between distance-based and weighted constraint based representation of preferences. With distances, the input consist of a set of goals and a predefined distance; with weighted formulas, it consists of a set of pairs (formula, weight). We now briefly show how both approaches relate. For the sake of clarity we give the transformation in the particular case of penalties ($*$ = sum)[10].

From distances to penalties: to a pair $\langle G, d \rangle$ we associate the set of weighted formulas $F = \{(G\ 1), (D(G,1)\ 1), \ldots (D(G,k)\ 1)\}$ where k is the largest integer such that $d(G,k) \neq \top$. Then we have

[10] A similar work could be done with other aggregation functions; it would however be more complicated and less natural, except in the case of *max*.

$\forall w$, $disu_F(w) = d(w, G)$. This transformation is polynomial when G is under DNF.

From penalties to distances: in this direction, the transformation is less easy and need the use of a weak weighted Hamming distance. Let $P = \{(\varphi_1\ \alpha_1), ..., (\varphi_n\ \alpha_n)\}$. We define $K' = K \cup \{\varphi_i \leftrightarrow v_i, i = 1 \ldots n\}$ where the v_i's are new propositional symbols, and let d the weak weighted Hamming distance corresponding to the weights $p_i = \alpha_i$ for each v_i, and 0 to all other propositional symbols. Then we have $\forall w \models K'$, $\sum_i d(w, v_i) = disu_{(K,P)}(w)$. This transformation is polynomial. It is however less general than in the other direction, because the distance used depends on the input.

4.2 Robustness

Suppose that a (individual or multi-agent) preference profile is "very consistent", i.e., after the aggregation is finished, there remains many possible worlds with a null disutility (i.e., satisfying all desires). Then, rather than picking any world from this set, we may choose to select one being *robust* in the sense that if new constraints or desires are added later on, this world has a chance to remain a good solution afterwards (a similar idea is the motivation for *supermodels* [11]). This amounts this time to computing discs *inside* (and not any longer *around*) a formula: the inner a world inside a formula, the further from the "boarder" of the formula and the more robust it is. Formally, we define $D(\varphi, -k) = K \wedge \neg D(\neg\varphi, k)$; we have $w \models D(\varphi, -k)$ iff $d \models K$ and $d(w, \neg\varphi) > k$. From what precedes we infer that $D(\varphi, -k)$ can be computed in polynomial time if $\neg\varphi$ is in DNF, i.e., if φ is in CNF. The *kernel* of a formula, defined by $Ker(\varphi) = D(\varphi, -k^*)$ where $k^* = \max\{k | D(\varphi, -k) \neq \bot\}$, contains the "most inner" (and thus preferred) worlds.

5 Related work and conclusion

5.1 Merging

Konieczny and Pino-Perez [14] propose some postulates for *merging* and show that an operator Δ (mapping beliefs sets, i.e. multisets of formulas, to knowledge bases) is a merging operator iff there exists an assignment δ mapping each knowledge set E to a preorder $\leq_E$ such that $Mod(\Delta(E)) = min(\leq_E)$. They show (Section 5 of [14]) that some specific merging operators can be induced by distances.

When we make the two following restrictions to our framework: (a) one goal only (G_i) for each agent i; (b) $K = \top$; then the following (easy) result holds:

Proposition 4 *For each distance d and each operator $\diamond$ [11], the operator $\oplus$ defined by $Mod(\oplus(G_1, \ldots, G_n)) = \{w \mid w$ optimal for $(G_1, ..., G_n)$ w.r.t. d and $\diamond\}$ $(= \{w \mid \diamond\{d(w, G_1), \ldots, d(w, G_n)\}$ minimal$\})$ is a merging operator.*

This result establishes a strong link between merging operators and distance-based logics. Using then the results of Section 3.1, we can establish a (less straightforward and more technical) link between merging and weighted logics.

5.2 Supermodels

The idea of selecting a *robust* model comes from Ginsberg et al. [11]. If a and b are integers, then w is a (a, b)-supermodel of φ iff for any consistent conjunction γ of at most a literals, then there exists a world w' such that $w' \models \varphi \wedge \gamma$ and $d(w, w') \leq b$. There is a strong connection with our discs around a formula and *supermodels*[11]. Namely, when $K = \top$, the following result holds:

Proposition 5
$w \models D(\varphi, k)$ iff w is a $(0, k)$-supermodel of φ, or, equivalently, iff w is not a $(k, 0)$-supermodel of $\neg\varphi$.

5.3 Distance-SAT, MAXSAT and computational issues

DISTANCE-SAT [2] is the following problem: "given a CNF formula φ, a partial world w and an integer k, does there exist a model w' of φ such that w' disagrees with w on at most k variables?". Clearly, the algorithms developed and experimented in [2] are a good starting point when it comes to design efficient algorithms for computing optimal decisions within the distance-based representation model. Similarly, algorithmic studies on MAXSAT (which can be seen as a particular case of weighted logics when $* = \diamond = +$ and all weights are unitary) and on *valued constraint satisfaction* [24] are a good starting point for computing optimal decisions in weighted logics.

Acknowledgments

We would like to thank the participants of the European working group "FUSION" supported by the European Commission for helpful discussions, with a special mention to Salem Benferhat and Isabelle Bloch. Thanks as well to Sébastien Konieczny, Pierre Marquis and Ramon Pino-Pérez.

[11] $*$ is useless because of assumption (a).

References

[1] K. Arrow. *Social Choice and Individual Values*, 2nd ed. (1st ed., 1951), John Wiley, 1963.

[2] O. Bailleux and P. Marquis. DISTANCE-SAT: COMPLEXITY AND ALGORITHMS. *Proceedings of AAAI'99*, p.642-647, 1999.

[3] I. Bloch and J. Lang. Towards mathematical morphologics. To appear.

[4] C. Boutilier. Towards a logic of qualitative decision theory. Proceedings of *KR'94*, p. 75-86, 1994.

[5] R. I. Brafman, N. Friedman. On decision-theoretic foundations for default. Proceedings of *IJCAI'95*, p. 1458-1465, 1995.

[6] M. Dalal. Investigations into a theory of knowledge base revision: preliminary report. Proceedings of *AAAI'98*, p. 475-479, 1988.

[7] J. Doyle and M. P. Wellman. Preferential semantics for goals. Proceedings of *AAAI'91*, p. 698-703, 1991.

[8] D. Dubois, J. Lang and H. Prade. Possibilistic logic. *Handbook of Logic in Artificial Intelligence and Logic Programming* (D.M. Gabbay, C.J. Hogger, J.A. Robinson, eds.), Vol. 3, 439-513, Oxford University Press. 1994.

[9] F. Dupin de Saint-Cyr. *Gestion de l'évolutif et de l'incertain en logiques pondérées* (in French). PhD thesis, Université Paul Sabatier, Toulouse, France, 1996.

[10] F. Dupin de Saint-Cyr, J. Lang and T. Schiex., Penalty logic and its link with Dempster-Shafer theory, *UAI'94*, 1994.

[11] M. L. Ginsberg, A. J. Parkes, A. Roy. Supermodels and Robustness. *Proceedings of AAAI'98*, p.334-339, 1998.

[12] P. Haddawy, S. Hanks, Representations for decision-theoretic planning: utility functions for deadline goals. Proceedings of *KR'92*, p. 71-82, 1992.

[13] S. Konieczny. *Sur la logique du changement : révision et fusion de bases de connaissances* (in French). PhD thesis, Université des Sciences et Technologies de Lille, 1999.

[14] S. Konieczny, R. Pino-Pérez, On the logic of merging, *KR'98*, p.488-498, 1998.

[15] C. Lafage, J. Lang. Logical representation of preferences for group decision making. Technical report, IRIT Université Paul Sabatier Toulouse. electronically available at `ftp://ftp.irit.fr/pub/IRIT/RPDMP/langlafa.ps.gz`

[16] J. Lang, Possibilistic logic: complexity and algorithms. To appear in *Handbook of Algorithms for Uncertainty and Defeasible Reasoning* (J. Kohlas, S. Moral, eds.), Kluwer Academic Publishers.

[17] J. Lang and P. Marquis. The complexity of weighted logics and related issues. In preparation.

[18] R.D. Luce, H. Raiffa, *Games and Decisions*, Wiley, New-York, 1967.

[19] H. Moulin, *Axioms of Cooperative Decision Making*, Cambridge University Press, Cambridge, 1988.

[20] C. H. Papadimitriou, *Computational complexity*, Addison-Wesley Publishing Company, 1993.

[21] G. Pinkas, Propositional nonmonotonic reasoning and inconsistency in symmetric neural networks. Proceedings of *IJCAI'91* , p.525-530, 1991.

[22] D. Poole. Decision-theoretic defaults. Proceedings of *UAI'92*, p. 190-197, 1992.

[23] M.V. Nagendra Prasad, S.E. Lander and V. R. Lesser. Cooperative learning over composite search spaces: experiences with a multi-agent design system. *Proceedings of AAAI'96*, 68-73.

[24] T. Schiex, H. Fargier and G. Verfaillie. Valued constraint satisfaction problems: hard and easy problems, Proceedings of *IJCAI'95*, p. 631-637, 1995.

[25] L. van der Torre and E. Weydert. Parameters for utilitarian desires in a qualitative decision theory. To appear in: *Applied Intelligence*.

[26] R.R. Yager, On ordered weighted averaging aggregation operators in multi-criteria decision making, *IEEE Transactions on Systems, Man and Cybernetics* 18, p.183-190, 1988.

Applications and Systems

OntoMorph: A Translation System for Symbolic Knowledge

Hans Chalupsky
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292 U.S.A.
hans@isi.edu

Abstract

A common problem during the life cycle of knowledge-based systems is that symbolically represented knowledge needs to be translated into some different form. Translation needs occur along a variety of dimensions, such as KR language syntax and expressivity, modeling conventions, representation paradigms, etc. As a tool to support the translation problem, we present the OntoMorph system. OntoMorph provides a powerful rule language to represent complex syntactic transformations, and it is fully integrated with the PowerLoom KR system to allow transformations based on any mixture of syntactic and semantic criteria. We describe OntoMorph's successful application as an input translator for a critiquing system and as the core of a translation service for agent communication. We further motivate how OntoMorph can be used to support knowledge base merging tasks.

1 Introduction

A common problem during the development of ontologies and knowledge-based systems is that symbolically represented knowledge needs to be translated into some different form. For example, integration of independently developed knowledge-based components [Cohen *et al.*, 1998], merging of overlapping ontologies [Valente *et al.*, 1999], communication between distributed, heterogeneous agents, or porting of knowledge-based systems to use a different knowledge representation infrastructure commonly require translation, since every encoding of knowledge is based on a multitude of representational choices and assumptions. Translation needs go well beyond syntactic transformations and occur along many dimensions, such as expressivity of representation languages, modeling conventions, model coverage and granularity, representation paradigms, inference system bias, etc., and any combination thereof.

Traditionally, such translations are either performed manually via text or knowledge base editors or via special-purpose translation software. Manual translation is slow, tedious, error-prone, hard to repeat and simply not practical for certain applications. Special-purpose translation software is difficult to write, hard to maintain and not easily reusable.

Being confronted with translation problems on a frequent basis, we are developing the OntoMorph system to facilitate ontology merging and the rapid generation of knowledge base (KB) translators. OntoMorph combines two powerful mechanisms to describe KB transformations: (1) *syntactic rewriting* via pattern-directed rewrite rules that allow the concise specification of sentence-level transformations based on pattern matching, and (2) *semantic rewriting* which modulates syntactic rewriting via (partial) semantic models and logical inference supported by an integrated KR system. The integration of these mechanisms allows transformations to be based on any mixture of syntactic and semantic criteria, which is essential to support the translation needs enumerated above. The OntoMorph architecture facilitates incremental development and scripted replay of transformations, which is particularly important during merging operations.

2 The Translation Problem

The general problem we are considering is shown in Figure 1: Given some source knowledge base KB_s we want to design a transformation function τ to transform it into a target knowledge base KB_t. A fundamental assumption in this formulation is that source

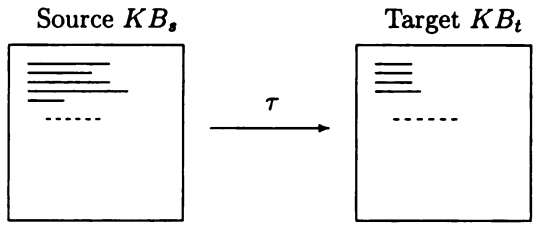


Figure 1: The knowledge translation problem

and target KBs are describable by a set of sentences in some linear, textual notation, where *sentence* means some independent syntactic unit as opposed to a well-formed logical formula associated with a truth value. This does not exclude graphical languages such as, for example, SNePS [Shapiro and Rapaport, 1992] or Conceptual Graphs [Sowa, 1992], since they usually also have some linear syntax to textually describe their networks. The translation does not necessarily have to span a whole knowledge base. In some cases, it might only involve single expressions.

A common correctness criterion for translation systems is that they preserve semantics, i.e., the meaning of the source and the translation has to be the same. This is not necessarily desirable for our transformation function τ, since it should be perfectly admissible to perform abstractions or semantic shifts as part of the translation. For example, one might want to map an ontology about automobiles onto an ontology of documents describing these automobiles. Since this is different from translation in the usual sense, we prefer to use the term *knowledge transformation* or *morphing*.

3 Dimensions of Mismatch

Despite the fact that the function τ might perform arbitrary semantic shifts, the most common scenario is to translate between different models of the same general domain. Unfortunately, these models can and in practice do differ along a multitude of dimensions. The most commonly encountered mismatches are outlined below.

KR language syntax: Every KR language comes with its own syntax, which is probably the most mundane but nevertheless annoying mismatch. For example, here are three different ways of defining automobiles as a subclass of road vehicles, one for Loom [MacGregor, 1991], a KL-ONE-style description logic, one for MELD, the representation language used by CYC [Lenat, 1995] and one for KIF [Genesereth, 1991]:

```
Loom:  (defconcept Automobile
         "The class of passenger cars."
         :is-primitive Road-Vehicle)

MELD: (#$isa #$Automobile #$Collection)
      (#$genls #$Automobile #$RoadVehicle)
      (#$comment #$Automobile
        "The class of passenger cars.")

KIF:  (defrelation Automobile (?x)
        "The class of passenger cars."
        :=> (Road-Vehicle ?x))
```

Apart from different surface syntax, there are also different *syntactic conventions* such as the spelling of names that are really part of the culture of the language users. For example, CYC names are mixed-case without hyphens as opposed to the hyphenated, case-insensitive spelling usually used with the other languages.

KR language expressivity: Every KR language trades off representational expressiveness with computational tractability. For example, negation, quantification, defaults, modal operators, representation of sets, etc. are supported by some languages and not by others. When translating between languages of different expressiveness, difficult choices have to be made in how to map certain representational idioms. For example, to represent that the typical capacity of a passenger car is five, we could use the following representation in Loom:

```
(defconcept Automobile
 :is-primitive Road-Vehicle
 :defaults (:filled-by passenger-capacity 5))
```

To represent the same in ANSI KIF which does not support defaults, one would have to resort to something like the following and then leave it up to some extra-logical means to properly reason with typicality assertions:

```
(defrelation Automobile (?x)
 :=> (and (Road-Vehicle ?x)
          (typical-passenger-capacity ?x 5)))
```

Modeling conventions: Even if the KR language and system for source and target KB are the same, differences occur because of the way a particular domain is modeled. For example, a choice one often has to make is whether to model a certain distinction by introducing a separate class, or by introducing a qualifying attribute relation. E.g., to distinguish between tracked and wheeled vehicles, one

could either introduce two subclasses of `Vehicle` called `Tracked-Vehicle` and `Wheeled-Vehicle`, or use an attribute relation as in `(traction-type My-Car wheeled)`. Which representation to choose is in most cases just a matter of taste or convention.

Model coverage and granularity: Models differ in their coverage of a particular domain and the granularity with which distinctions are made. This is often the very reason why ontologies are merged. For example, one ontology might model cars but not trucks. Another one might represent trucks but only classify them into a few categories, while a third one might make very fine-grained distinctions between types of trucks based on their general physical structure, weight, purpose, etc.

Representation paradigms: Different paradigms are used to represent concepts such as time, action, plans, causality, propositional attitudes, etc. For example, one model might use temporal representations based on Allen's interval logic [Allen, 1984], while another might use a representation based on time points. Section 5 describes a situation where two different representations of "purpose" had to be reconciled with the help of OntoMorph.

Inference system bias: Last but not least, another reason why models often look a certain way is that they were constructed to produce desired inferences with a particular inference engine or theorem prover. For example, in a description logic such as Loom, certain inferences are well-supported by the classifier, while others are only supported at the instance or individual level. This trade-off can influence one's choice whether to model something as a class or as an individual. See [Valente *et al.*, 1999] for a discussion of modeling examples exhibiting inferencing bias.

4 OntoMorph

To facilitate the rapid specification of KB transformation functions such as τ described above, OntoMorph combines two powerful mechanisms: (1) *syntactic rewriting* via pattern-directed rewrite rules that allow the concise specification of sentence-level transformations based on pattern matching, and (2) *semantic rewriting* which modulates syntactic rewriting via (partial) semantic models and logical inference.

4.1 Syntactic Rewriting

To allow translation between arbitrary KR languages that can differ widely in their syntax, expressiveness, and underlying knowledge model, OntoMorph uses *syntactic rewriting* as its core mechanism. Input expressions are first tokenized into lexemes and then represented as syntax trees whose subtrees represent parenthesized groups (similar to Lisp s-expressions). The tree structure exists only logically; a tree is represented internally as a flat sequence of tokens.

For example, the expression `f(g([x],y))` would be represented by the token sequence

`'f' '(' 'g' '(' '[' 'x' ']' ',' 'y' ')' ')'`

which, logically, represents the syntax tree shown in Figure 2. The significance of the tree structure is that complete subtrees can be matched by a single pattern variable, and that sequence variables do not consume tokens beyond subtree boundaries.

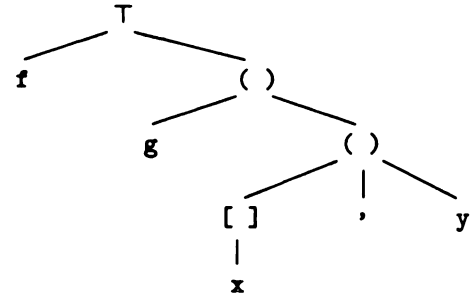


Figure 2: Syntax tree representation of `f(g([x],y))`

OntoMorph's syntactic rewrite rules have this general form:

$$pattern \Rightarrow result$$

The left-hand-side pattern matches and destructures one or more syntax trees while the right-hand side generates new trees of the desired format by explicitly specifying new structure, reassembling some of the destructured information and by possibly further rewriting some subexpressions. For example, a very simple rule to convert a MELD type assertion into its Loom analogue would look like this (pattern variables are prefixed with a `'?'`):

```
(isa ?x ?class) ==> (tell (?class ?x))
```

The ability to describe such transformations in a very direct and concise fashion was an important design objective for OntoMorph. When researching the relevant parsing and pattern-match literature and technology, we found that a language called Plisp (or Pattern Lisp) [Smith, 1990], which in turn is a direct descendent of the Lisp 70 pattern matcher [Tesler *et al.*, 1973], came

closest to our intuitions on how such transformations should be represented and executed. Unfortunately, none of these systems is alive and well anymore, so we had to develop our own version.

4.1.1 Pattern Language

OntoMorph's pattern language and execution model is strongly influenced by Plisp, even though the actual surface syntax is quite different. The pattern language can match and destructure arbitrarily nested syntax trees in a direct and concise fashion. A short overview of the available constructs is given below:

Literals such as `foo`, `"bar"`, `42`, `(, (a (b c))`, etc., have to be matched by identical literal tokens (or token sequences).
Variables (indicated by a `?`-prefix), e.g., `?x`, `?why` or the anonymous variable `?`, can match individual input tokens such as `foo` or a token sequence representing a tree such as `(a (b c))`. Once a variable is bound, it can only be matched by literal tokens matching its binding.
Sequence variables (indicated by a `??`-prefix), e.g., `??h`, `??tail` or the anonymous variable `??`, can match tree subsequences such as `c (d)` in the tree `(a b c (d))`. For example, the pattern `(??x b ??y)` matches the tree `(a b c (d))` by binding `??x` to the single-element sequence `a` and `??y` to the sequence `c (d)`. Sequence variables cannot consume tokens beyond subtree boundaries.
Grouping (expressed via braces) defines compound patterns. For example, the pattern `{a ?x c}` can match the token sequence `a b c`. Groups are also used to apply pattern modifiers such as repetition to compound patterns.
Alternatives (expressed via vertical bars) define disjunctive patterns such as `{a | (b ?x) | c d}`. The pattern matches if one of its components succeeds.
Optionals such as `{a b [c]}` are syntactic sugar for the more verbose `{a b | a b c}` notation.
Repetition (expressed with the usual `*` or `+` notation) indicates that a pattern can be matched multiple times. As a generalization, an `m-n` range can be supplied to mandate that there have to be at least `m` and at most `n` matches. For example, `{a | b}+` matches any sequence of `a`'s and `b`'s with length ≥ 1, `{a | b}*1-2` matches only those sequences with lengths between 1 and 2.
Input binding binds the input matched by a complex pattern to a single variable. This is useful if a pattern has alternatives and it is necessary to refer to what was actually matched by it in the right-hand side of a rewrite rule (without alternatives, the same could be achieved by literally repeating the pattern). For example, `?x := {a | (b ?y) |c}` matched against `(b d)` binds `?x` to `(b d)`.

Below is an example pattern that combines various of the elements described above to match and destructure a Loom concept definition (note, that the example only covers some aspects of the Loom concept language). The alternatives in combination with the repetition construct allow the keyword/value pairs to appear in any order. The construction for the `:annotations` keyword extracts a documentation string (which might appear in various ways) while ignoring everything else:

```
(defconcept ?name
  {?is := {:is | :is-primitive} ?def |
   :characteristic ?c |
   :annotations
    ?a := {(documentation ?d) |
           (:and ?? (documentation ?d) ??) |
           ?}}*0-3)
```

The pattern matches and destructures concept definitions such as this one:

```
(defconcept Dog
   :annotations
      (:and Class (documentation "Canine"))
   :is-primitive Animal)
```

4.1.2 Execution Model

Rewrite rules are applied according to the following simple execution model: Initially, an input stream is constructed consisting of the token sequence representing the input expression. When a rewrite rule is applied, its left-hand-side pattern consumes tokens from the input stream by matching them against the elements of the pattern. If the pattern succeeds, the right-hand-side result is assembled and the resulting tokens are pushed back onto the input stream where they replace the consumed input and become available as input to further rewrite rules. For example, assume we have the following input stream:

```
'(' 'isa' 'car1' 'Ford' ')' '(' ...
```

Now we apply the type transformation rule from before:

```
(isa ?x ?c) ==> (tell (?c ?x))
```

Applying the rule modifies the input stream to the following:

```
'(' 'tell' '(' 'Ford' 'car1' ')' ')' '(' ...
```

Assembly of a rule result involves collecting its right-hand-side component tokens from left to right into

a temporary store. Literal tokens such as the `tell` above are simply copied, variables are substituted by their bindings, and functions and recursive rule invocations (explained below) are evaluated and their results collected. Once all right-hand-side components have been successfully evaluated, the content of the temporary store is prepended to the input stream where it replaces the input consumed by the left-hand-side.

Rewrite rules are always assembled into rule sets of the following form:

(defruleset *name*
  $pattern_1 \Rightarrow result_1$
  ...
  $pattern_n \Rightarrow result_n$)

The individual rules are implicitly OR-ed and tried in sequence. The ruleset succeeds with the result of the first successful element rule.

Explicit invocation of named rulesets is the primary mechanism to achieve recursion, which is necessary to handle the translation of recursive structures. Apart from this computational aspect, grouping rules improves modularity, and it also greatly improves efficiency, since it restricts the set of rules tried to rewrite any given subexpression.

While matching a pattern and also during the assembly of the right-hand-side result which might involve further rewrites, a rule may fail. In that case execution backtracks to the most recent match choice point. After all input has been consumed and no more rules need to be applied, the process terminates and the resulting state of the input stream constitutes the result of the rewrite operation which is then either printed to some storage medium or used directly as part of a KB operation such as assertion or retrieval.

4.1.3 Function Calls and Rule Invocations

To allow the parsing and rewriting of recursive structures, other rulesets as well as built-in functions can be invoked explicitly anywhere in a pattern. Such invocations are written with an angle bracket syntax to distinguish them from the regular syntax tree notation. For example, the call `<Term ?x>` invokes a function or ruleset called `Term` on the argument `?x`. Before the function is called, its arguments are evaluated and the results pushed back onto the input stream from which they are then consumed. Excess or missing arguments are left on or filled in from the remainder of the input. When a function or ruleset invocation on the left-hand side of a rule returns, its result gets pushed back onto the input where it immediately becomes available to subsequent pattern elements. On the right-hand side (as described above), the result gets first collected in a temporary store until all right-hand-side tokens of the rule have been evaluated.

The following two rule sets constitute a simple transformation system for arithmetic expressions (note, that the + and * symbols need to be escaped to treat them as ordinary characters):

```
(defruleset Term
  (?op := {\+ | - | \* | /} ?x ?y)
     ==> (?op <Term ?x> <Term ?y>)
  (1\+ ?x) ==> (\+ <Term ?x> 1)
  (1- ?x) ==> (- <Term ?x> 1)
  (square ?x) ==> (\* <Term ?x> <Term ?x>)
  ?x ==> ?x)

(defruleset Condition
  (lt ?x ?y)
     ==> (negative? (- <Term ?x> <Term ?y>))
  (gt ?x ?y) ==> <Condition (lt ?y ?x)>)
```

To apply these rules, we can use the OntoMorph function `rewrite` which takes an input expression and a start rule as arguments. For example,

```
(rewrite (gt (/ (1+ M) N) (square N))
         Condition)
```

returns the following result:

```
(negative? (- (* N N) (/ (+ M 1) N)))
```

Currently, OntoMorph uses a Lisp-style reader to tokenize the input into individual lexemes. Future versions will allow the specification of customized tokenizers in order to support the translation of languages with different lexical conventions.

4.2 Semantic Rewriting

Syntactic rewriting is a powerful mechanism to describe pattern-based, sentence-level transformations. However, it is not sufficient if the transformations have to consider a larger portion of the source KB, possibly requiring logical inference. A simple example of such a transformation is conflation. Suppose one wants to conflate all subclasses of `Truck` occurring in some ontology about vehicles into a single `Truck` class. This involves among other things the rewriting of all type assertions involving trucks. Using syntactic rewriting alone, one would need a rule such as the following which explicitly lists all subtypes of `Truck`:

```
(defruleset Conflate-Truck-Types
```

```
({Light-Truck | Heavy-Truck | ...} ?x)
  ==> (Truck ?x))
```

For large taxonomies this is of course neither elegant nor feasible. Instead of the purely syntactic test based on truck class names, a semantic test is needed to check whether a particular class is a subclass of `Truck`.

To facilitate the utilization of semantic information, OntoMorph is built on top of the PowerLoom knowledge representation system. PowerLoom is a successor to the Loom system that supports definitions and rules in a typed variant of KIF combined with a powerful inference engine and a classifier. Wherever a function call is legal in a rewrite rule, a PowerLoom function can be called to change or access the state of the current KB.

One way to solve the conflation problem is to establish a partial mirror of the source KB within an intermediate PowerLoom KB. This can be done with a specialized set of rewrite rules that import source sentences representing taxonomic relationships, but ignoring all other information, for example, by only paying attention to **subset** and **superset** assertions. This step can be viewed as the first pass of a two-pass translation scheme. In the second pass, the actual translation rules are applied, but now they can also access the semantic information established in the first pass. Making use of the imported taxonomic knowledge, the following rule can conflate all truck types:

```
(defruleset Conflate-Truck-Types
 {(?class ?x) <ask (subset-of ?class Truck)>}
   ==> (Truck ?x))
```

The left-hand-side contains a group of patterns which is treated as a conjunction. The first conjunct (`?class ?x`) simply matches any type assertion. The second one calls **ask** which triggers a PowerLoom query. Note that `?class` will be substituted with the matched class name, thus, the query will be fully ground. Since **ask** is a boolean-valued function, its result will simply be treated as a test instead of being pushed back onto the input stream.

Using semantic import rules, an arbitrarily precise image of the source KB semantics can be established within PowerLoom (limited only by the expressiveness of first-order logic). Then syntactic rewrite rules can use the imported semantic information to perform rewrites based on any mixture of syntactic and semantic criteria.

Obviously, the precision of the semantic import will affect the quality of the translation. For example, in the scenario above the semantic import only considered **subset** and **superset** assertions. Depending on the nature of the source KB, there might be other information and rules that would allow one to infer additional taxonomic relationships. These would then not be inferable within the partial PowerLoom mirror KB which might adversely affect the translation quality.

Whether this is a problem and how to best solve it has to be decided on a case-by-case basis. One solution is to use PowerLoom as an interlingua and import everything from the source KB (again, this is limited only by the expressiveness of PowerLoom). The disadvantage of this scheme is that one effectively needs two sets of translation rules, one to translate from the source into PowerLoom, and one to go from PowerLoom to the target representation. Alternatively, it might be possible to call out to the KR system that has the source KB loaded and use its inferencing capabilities directly. This can either be done via some special-purpose API, or, if supported, via a protocol such as OKBC [Chaudhri *et al.*, 1998]. Which route to take will depend on a variety of pragmatic factors. For the OntoMorph applications constructed to date, importing partial semantic information into PowerLoom was sufficient to support all rewriting needs.

5 OntoMorph Applications

OntoMorph has already been successfully applied in a couple of domains. One involved the translation of military scenario information for a plan critiquing system. In the second it formed the core of an agent translation service called Rosetta, where it was used to translate messages between two communicating planning agents that used different representations for goals.

5.1 Course of Action Critiquer

One of the challenge problems that drove the second phase of DARPA's High Performance Knowledge Bases (HPKB) project [Cohen *et al.*, 1998] was to develop critiquing systems for military courses of actions (or COAs) which are high-level, plan-like descriptions of military operations. To represent a particular COA, scenario information from a graphical sketch pad was fused with information from a natural language description of the COA by a program called the Fusion engine. The combined description of the COA was represented in CYC's MELD language and then fed to five independent critiquing systems built by different teams. Only one of the critiquers was using CYC directly and did not have to translate the Fusion engine output. All others had to use some form of translation system. Many different scenarios had to be handled in

a tight evaluation schedule, thus, manual translation was not an option. OntoMorph was chosen to translate the Fusion output for the critiquer based on the EXPECT knowledge acquisition system [Gil, 1994] which uses Loom to represent its knowledge. What follows is a list of translation issues that arose, and how they were solved:

Different Names: While most of the names generated by the Fusion engine were shared by the EXPECT critiquer, some of them differed due to parallel independent development of critiquers and ontologies as well as personal style. Renaming was taken care of with simple rules like the following:

```
(DEFRULESET rename-collection
  Fix-MilitaryTask ==> FIX
  {ProtectingSomething |
   ProtectingPhysicalRegion} ==> PROTECT
  Translation-LocationChange ==> MOVE
  ...)
```

Different Syntax: OntoMorph started with a KIF translation of the Fusion output which still contained various MELD idioms that needed to be translated into Loom syntax. For example, `isa` assertions such as `(isa task1 Fix-MilitaryTask)` had to be translated into the Loom idiom `(FIX task1)` (which here also involved a name change). MELD frame predicates were also easily translated into Loom with the following rule:

```
(DEFRULESET rewrite-frame-predicate
  (relationInstanceExistsCount
    ?relation ?instance ?type ?count)
  ==> (:ABOUT <rewrite-term ?instance>
        (:EXACTLY ?count
          <rename-relation ?relation>
          <rename-collection ?type>)))
```

Different Representations: The most challenging difference to overcome was the different representations used to represent the purposes of tasks. The Fusion engine used an idiom that related a task with a proposition whose truth was supposed to be brought about by carrying out the task. For example, to state that the purpose of the task carried out by `BlueDivision1` was to protect `Boundary1`, the following representation was used:

```
(taskHasPurpose BlueDivisionTask
  (thereExists ?p
    (isa ?p
         (CollectionSubsetFn
           ProtectingSomething
           (TheSetOf ?obj
             (and (objectTakenCareOf
                    ?obj Boundary1)
                  (performedBy
                    ?obj BlueDivision1)))))))
```

This can roughly be paraphrased as follows: The purpose of `BlueDivisionTask` is to bring about the existence of an event `?p` that is an instance of the event type `ProtectingSomething` restricted by the set of events in which `BlueDivision1` takes care of `Boundary1` (the restriction is expressed via the `CollectionSubsetFn` construction). This representation goes far beyond the expressiveness of Loom which does not have a way to represent higher-order sentences such as the above. It also did not meet the requirements of the EXPECT critiquer, which needed a reified purpose representation such as the following:

```
(AND (PROTECT protect00)
     (PURPOSE-ACTION protect00)
     (PURPOSE-OF BlueDivisionTask protect00)
     (ACTION-OBJ protect00 Boundary1)
     (WHO protect00 BlueDivision1))
```

The final version of the Fusion engine only used three structurally different purpose representation patterns. Each of them could be handled by an OntoMorph rule such as the following:

```
(DEFRULESET rewrite-purpose-pattern1
  {(taskHasPurpose ?task
     (thereExists ?var
       (isa ?var
            (CollectionSubsetFn
              ?type
              (TheSetOf ?action ?body)))))
   <generate-unique-name
     <rename-collection ?type>>
   ?purpose}
  ==> (AND
        (<rename-collection ?type> ?purpose)
        (PURPOSE-ACTION ?purpose)
        (PURPOSE-OF ?task ?purpose)
        <rewrite-purpose-setof-body
          ?body ?action ?purpose>))
```

The mapping between the two representations is very direct and makes good use of OntoMorph's destructuring facilities for syntax trees. The only complication is the extra right-hand-side function call to create a skolem individual needed to represent the reified purpose. This is taken care of by a call to the built-in function `generate-unique-name` which bases the generated name on the supplied argument (in this case,

the renamed base event type). It does not consume anything from the input stream but simply pushes the result back onto it where it is then consumed by the ?purpose variable.

Missing Representations: Some information needed by the EXPECT critiquer such as COA substructure and task/subCOA associations was not explicitly represented and needed to be recovered by some of OntoMorph's semantic rewrite features, e.g., by keying in on "meta-information" such as where in the Fusion output certain assertions were made.

For example, to associate a task with a particular sub-COA, it was necessary to track what tasks were performed by what unit which was handled by the following two rules:

```
(DEFRULESET track-COA-assertion
  (unitAssignedToTask ?task ?unit)
  ==> <!ASSERT
        (AND (Term ?task) (Term ?unit)
             (unitAssignedToTask
               ?task ?unit))>)

(DEFRULESET get-task-assigned-to-unit
  {?unit
   <@RETRIEVE \?t
     (= (unitAssignedToTask \?t) ?unit)>
   ?task}
  ==> <OBJECT-NAME ?task>)
```

The first rule creates a PowerLoom assertion for each `unitAssignedToTask` statement in the Fusion scenario. PowerLoom expects all its objects to be typed before they are used which is the reason for the additional `Term` assertions. The second rule retrieves the task recorded for a particular unit which was then used to associate it with the sub-COA in which the particular unit was involved. Note, that the `?t` variable within the PowerLoom `retrieve` statement is escaped, since it is a retrieval variable and not a pattern variable of the rewrite rule. The `?unit` variable, however, is a pattern variable, thus, its binding is substituted before the retrieval is executed and is seen by PowerLoom as an ordinary constant.

The complete translator was comprised of about 30 rulesets, 10 of which were necessary just to track unrepresented COA structure. The size of the translator was about 15 kilobytes of text.

5.2 Rosetta Agent Translation Service

Rosetta is a prototype of an ontology-based translation service operating in a domain of planning agents. It reengineers some aspects of a technology integration effort described in [Cox and Veloso, 1997] which connects the ForMAT case-based planning tool [Mulvehill and Christey, 1995] with the Prodigy/Analogy planner [Veloso, 1994; Veloso *et al.*, 1995]. In the original experiment, special-purpose translators were constructed to allow ForMAT and Prodigy/Analogy to communicate. Rosetta is an attempt to show how these translators can be replaced with a more flexible, general-purpose translation architecture that promotes reuse and that can scale up to large numbers of heterogeneous, communicating agents. The full motivation and details of the Rosetta architecture are given in [Blythe *et al.*, 2000]. Here we will only touch on some aspects and how they are handled by the OntoMorph system.

The main idea behind Rosetta is that it provides a representation interlingua in conjunction with a repository of broad-coverage as well as domain-specific ontologies that can be used to represent content expressions exchanged by heterogeneous communicating agents. Each agent is associated with a wrapper that (1) translates its message content language into the interlingua used by Rosetta, and (2) if necessary, aligns terms of the agent ontology with Rosetta's ontologies. Within Rosetta, each agent is associated with a model that represents relevant aspects of the agent's domain. As motivated in Section 3, using the same KR language and system to model a domain does not by itself eliminate the need for translation, since different representations can be used to express the same semantic content. To facilitate translations between such different representations, Rosetta has a library of representation reformulation rules.

To translate a message between agents A and B, agent A first uses its wrapper to translate the message content into Rosetta's format and sends it to Rosetta. Rosetta then checks whether any reformulation rules need to be applied to make the message understandable by agent B, and, if so, applies them. The resulting message is then sent to agent B which uses its own wrapper to translate it into its internal format. One of the advantages of this architecture is that the portion of the necessary translation mappings encodable in the wrappers grows only linearly with the number of different agent classes.

The uses of OntoMorph within this scenario were twofold: (1) It provided an obvious solution to implement the agent wrappers by primarily relying on its syntactic rewriting features. (2) Its semantic rewriting features were used to implement the necessary repre-

sentation reformulation rules. For example, the following top-level rule was used to translate a goal posted by the ForMAT planning tool into the Rosetta representation (note, that only the most relevant aspects of these rules are reproduced to save space):

```
(defruleset format-to-rosetta-wrapper
  {(:goal ?goal)
   <translate-format-goal-to-rosetta ?goal>
   ?translated-goal}
 ==>
  (message
    (content
      (find (object plans)
            (for (Objective-Based-Goal
                   ?translated-goal))))
    ...) ...)
```

This rule translates a ForMAT request such as

```
(:goal
  (G-144 :Send-Hawk
         ((force 42nd-Batt)
          (geographic-location Big-Town))))
```

into the following representation understandable by Rosetta (the message content language used for this prototype is based on the verb clause goal language used by the EXPECT system):

```
(message
  (content
    (find (object plans)
          (for (Objective-Based-Goal
                 (send-unit
                   (object 42nd-Batt)
                   (to Big-Town))))))
  ...)
```

Once this message arrives at Rosetta, it is handled by its top-level translation rule whose main purpose it is to trigger the translation of content expressions:

```
(defruleset translate-rosetta-message
  {(message
    {(content ?content) |
     ...}*4-4)
   <map-performative ?content>
   ?mapped-performative}
 ==>
  (message
    (content ?mapped-performative)
    ...) ...)
```

One of the interesting aspects of the communication between ForMAT and Prodigy/Analogy is that ForMAT uses an objective-based or verb-centered representation such as "send troops to X" to represent its goals. Prodigy/Analogy, on the other hand, needs to be given a state-based goal representation such as "troops deployed at X" to generate a plan. To be able to represent these different kinds of goals as well as other planning-related aspects, Rosetta employed the PLANET ontology developed by Blythe and Gil [Blythe *et al.*, 2000]. To translate between objective-based and state-based goals, Rosetta uses a (heuristic) reformulation rule that looks for the primary effect of the planning operator describing the objective-based goal to serve as its state-based translation. Here are two of the central reformulation rules involved in this mapping:

```
(defruleset map-objective-to-state-based-goal
  {?goal-instance ??roles
   <find-equivalent-operator ?goal-instance>
   ?operator
   <get-primary-effect ?operator> ?effect}
 ==>
  (State-Based-Goal
    <map-operator-and-roles
      ?operator ?effect (??roles)>))

(defruleset find-equivalent-operator
  {?goal-instance
   <@most-specific-named-descriptions
     <retrieve-tuples all \?op
      (and (member-of ?goal-instance \?op)
           (context-of
             \?op <get-agent-model
                     <current-receiver>>)
           (exists \?effect
             (role-type primary-effects
                        \?op \?effect)))>>
   ?equiv-operator}
 ==>
  <object-name ?equiv-operator>)
```

The first thing Rosetta does is to find a planning operator in its model of the Prodigy/Analogy agent that is a suitable match for the operator requested by ForMAT. It does so by looking for the most specific operator description that matches the description of the goal posted by ForMAT by using a PowerLoom subsumption test. After the operator is found, its primary effect is used as a state-based goal description that can be passed on to Prodigy. Once the top-level `translate-rosetta-message` rule terminates, the translated message looks like this and is sent to Prodigy:

```
(message
```

```
(content
  (find (object plans)
        (for (State-Based-Goal
               (is-deployed
                 (object 42nd-Batt)
                 (at Big-Town))))))
...)
```

Finally, the Prodigy/Analogy wrapper translates that into the following which can be sent directly to the planner:

```
(:find-plans
   (is-deployed 42nd-Batt Big-Town))
```

The Rosetta application provides a nice testbed for all aspects of OntoMorph. Syntactic rewriting is exercised in the agent wrappers, semantic rewriting is exercised to perform representation reformulations, and a mixture of both controls the scripting of the overall translation process. Furthermore, the tight integration with the PowerLoom KR system and the interpreted nature of the rewrite rules provide for a very productive, incremental development cycle.

6 Related Work

Ontolingua [Gruber, 1993; Fikes *et al.*, 1997] is an attempt to avoid the translation problem by providing a centralized ontology repository that encourages reuse, and an ontology specification language that serves as an interlingua whose representational primitives can be translated into a variety of target KR languages by special-purpose translators. However, since the generated translations cannot be controlled, modifications such as changing modeling conventions or performing semantic shifts is not possible. While avoiding translation is always a good strategy, it is not always possible such as in the case of distributed, heterogeneous agents. Using one big, centralized ontology as done by CYC has similar drawbacks. In particular, it becomes problematic when a smaller system that only relies on a portion of the ontology needs to be fielded. Another alternative to translation is the use of lifting axioms as done in [Frank *et al.*, 1999]. Lifting axioms can only be used in systems expressive enough to support them. Another drawback is that they perform translations via logical inference at query time, which could be prohibitively expensive.

Since part of OntoMorph can be viewed as a parser specification system, it is legitimate to ask how it compares to other parsing technology such as YACC, definite clause grammars, natural language parsers such as ATNs, etc. YACC parsers are only applicable to context-free languages that are LR(1), which is too restrictive for a general-purpose translation system. Natural language parsers such as ATNs could in principal be used to implement a rewrite system, but since they are geared towards parsing of natural language sentences instead of arbitrary syntax trees, the specification would be less direct and more difficult. Definite clause grammars probably come closest to our desiderata for direct and concise specification of transformation rules, however, extra support would be necessary to support certain conveniences of the OntoMorph pattern language such as sequence variables and bounded repetition of compound patterns. Additionally, the integration with a KR system such a PowerLoom would still be missing which is a crucial part of OntoMorph's utility. Similar objections hold for languages such as POP-11 which already provide some of the pattern match functionality needed by OntoMorph, but lack the combination of features and the integration with a KR system such as PowerLoom.

7 Future Work: Ontology Merging

One of the primary motivations for the development of OntoMorph was to support merging of overlapping ontologies. Merging two or more source ontologies into a merged ontology involves the following steps:

1. Finding semantic overlap or *hypothesizing alignments.*
2. Designing transformations to bring the sources into mutual agreement.
3. Editing or *morphing* the sources to carry out the transformations.
4. Taking the union of the morphed sources.
5. Checking the result for consistency, uniformity, and non-redundancy and if necessary repeating some or all of the steps above.

These steps have different degrees of difficulty and are supported to various degrees by the state of the art. For example, techniques for hypothesizing alignments have been developed during large-scale ontology merging tasks as described in [Knight and Luk, 1994; Hovy, 1998; McGuinness *et al.*, 2000], and consistency checking is already fairly well supported by today's KR systems. Designing the necessary transformations is probably the most difficult and least automatable task, since it involves understanding the meaning of the representations. Additionally, this step often involves human negotiation to reconcile competing views on how a particular modeling problem should be solved.

At the center of every merging operation is step 3, since before ontologies can be merged they have to be transformed into a common format with common names, common syntax, uniform modeling assumptions, etc., which always involves some of the transformation operations described in Section 3. Since merging is an iterative process, it is very important that these transformations can be specified easily and carried out repeatedly and automatically with a tool such as OntoMorph. This is even more important in the context of tracking changes to one of the sources in a later re-merge. Without a clear and executable specification of the transformations used in the initial merge, much of the merging work has to be redone by hand. By using a tool such as OntoMorph, many of the necessary transformation rules will be reusable as is, and only the changed and extended portions of the modified source ontology will require adapted or new rewrite rules.

8 Conclusion

We presented the OntoMorph translation system for symbolic knowledge. OntoMorph provides a powerful rule language to represent complex syntactic transformations, and it is fully integrated with the PowerLoom KR system to allow transformations based on any mixture of syntactic and semantic criteria. OntoMorph facilitates the direct and concise description of transformations to solve knowledge translation needs along a multitude of dimensions. OntoMorph's has been successfully applied as an an input translator for a critiquing system and as the core of a translation service for communication among heterogeneous agents.

Acknowledgements

This research was supported by the Defense Advance Research Projects Agency under Air Force Research Laboratory contracts F30602-97-1-0194 and F30602-97-1-0195. Many thanks to Yolanda Gil and Bob MacGregor for their comments on earlier versions of this paper.

References

[Allen, 1984] J.F. Allen. Toward a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[Blythe *et al.*, 2000] J. Blythe, Y. Gil, H. Chalupsky, and R.M. MacGregor. Supporting translation among planning agents. Internal Project Report, USC Information Sciences Institute, 2000.

[Chaudhri *et al.*, 1998] V.K. Chaudhri, A. Farquhar, R. Fikes, P.D. Karp, and J.P. Rice. OKBC: A programmatic foundation for knowledge base interoperability. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, pages 600–607, Menlo Park, July 26–30 1998. AAAI Press.

[Cohen *et al.*, 1998] P.R. Cohen, R. Schrag, E. Jones, A. Pease, A. Lin, B. Starr, D. Easter, D. Gunning, and M. Burke. The DARPA High Performance Knowledge Bases project. *Artificial Intelligence Magazine*, 19(4):25–49, 1998.

[Cox and Veloso, 1997] M.T. Cox and M.M. Veloso. Controlling for unexpected goals when planning in a mixed-initiative setting. In E. Costa and A. Cardoso, editors, *Proceedings of the Eighth Portuguese Conference on Artificial Intelligence (EPIA-97)*, volume 1323 of *LNAI*, pages 309–318, Berlin, 1997. Springer.

[Fikes *et al.*, 1997] R. Fikes, A. Farquhar, and J. Rice. Tools for assembling modular ontologies in Ontolingua. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97)*, pages 436–441, Menlo Park, July 27–31 1997. AAAI Press.

[Frank *et al.*, 1999] G. Frank, A. Farquhar, and R. Fikes. Building a large knowledge base from a structured source. *IEEE Intelligent Systems*, 14(1):47–54, 1999.

[Genesereth, 1991] M.R. Genesereth. Knowledge interchange format. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 599–600, San Mateo, CA, USA, April 1991. Morgan Kaufmann Publishers.

[Gil, 1994] Y. Gil. Knowledge refinement in a reflective architecture. In *Proceedings of the 12th National Conference on Artificial Intelligence. Volume 1*, pages 520–526, Menlo Park, CA, USA, July 31–August 4 1994. AAAI Press.

[Gruber, 1993] T.R. Gruber. A translation approach to portable ontology specifications. *Knowledge Aquisition*, 5(2):199–220, 1993.

[Hovy, 1998] E.H. Hovy. Combining and standardizing large-scale, practical ontologies for machine translation and other uses. In *Proceedings of the*

First International Conference on Language Resources and Evaluation (LREC), Granada, Spain, 1998.

[Knight and Luk, 1994] K. Knight and S.K. Luk. Building a large-scale knowledge base for machine translation. In *Proceedings of the 12th National Conference on Artificial Intelligence. Volume 1*, pages 773–778, Menlo Park, CA, 1994. AAAI Press.

[Lenat, 1995] D. Lenat. CYC: A Large Scale Investment in Knowledge Infrastructure. *Communications of the ACM*, 38(11):32–38, November 1995.

[MacGregor, 1991] R.M. MacGregor. Inside the LOOM description classifier. *ACM SIGART Bulletin*, 2(3):70–76, 1991.

[McGuinness *et al.*, 2000] D.L. McGuinness, R.E. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In A.G. Cohn, F. Giunchiglia, and B. Selman, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000)*, San Francisco, CA, 2000. Morgan Kaufmann.

[Mulvehill and Christey, 1995] A. Mulvehill and S. Christey. *ForMAT – a Force Management and Analysis Tool.* MITRE Corporation, Bedford, MA, 1995.

[Shapiro and Rapaport, 1992] S. C. Shapiro and W. J. Rapaport. The SNePS family. *Computers & Mathematics with Applications*, 23(2–5):243–275, January–March 1992.

[Smith, 1990] D.C. Smith. *Plisp User's Manual.* Apple Computer, August 1990.

[Sowa, 1992] J. F. Sowa. Conceptual graphs as a universal knowledge representation. In Fritz Lehmann, editor, *Semantic Networks in Artificial Intelligence*, pages 75–93. Pergamon Press, Oxford, 1992.

[Tesler *et al.*, 1973] L. Tesler, H. Enea, and D.C. Smith. The Lisp70 pattern matching system. In Nils J. Nilsson, editor, *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, pages 671–676, Standford, CA, August 1973. William Kaufmann.

[Valente *et al.*, 1999] A. Valente, T.A. Russ, R.M. MacGregor, and W.R. Swartout. Building and (re)using an ontology of air campaign planning. *IEEE Intelligent Systems*, 14(1):27–36, 1999.

[Veloso *et al.*, 1995] M. Veloso, J. Carbonell, A. Pérez, D. Borrajo, E. Fink, and J. Blythe. Integrating planning and learning. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1), 1995.

[Veloso, 1994] M.M. Veloso. *Planning and learning by analogical reasoning*, volume 886 of *LNAI*. Springer, New York, NY, 1994.

An Environment for Merging and Testing Large Ontologies

Deborah L. McGuinness
Knowledge Systems Laboratory
Computer Science Department
Stanford University
dlm@ksl.stanford.edu

Richard Fikes
Knowledge Systems Laboratory
Computer Science Department
Stanford University
fikes@ksl.stanford.edu

James Rice
CommerceOne
Mountain View, CA
rice@jrice.com

Steve Wilder
Knowledge Systems Laboratory
Computer Science Department
Stanford University
wilder@ksl.stanford.edu

Abstract

Large-scale ontologies are becoming an essential component of many applications including standard search (such as Yahoo and Lycos), e-commerce (such as Amazon and eBay), configuration (such as Dell and PC-Order), and government intelligence (such as DARPA's High Performance Knowledge Base (HPKB) program). The ontologies are becoming so large that it is not uncommon for distributed teams of people with broad ranges of training to be in charge of the ontology development, design, and maintenance. Standard ontologies (such as UNSPSC) are emerging as well which need to be integrated into large application ontologies, sometimes by people who do not have much training in knowledge representation. This process has generated needs for tools that support broad ranges of users in (1) merging of ontological terms from varied sources, (2) diagnosis of coverage and correctness of ontologies, and (3) maintaining ontologies over time. In this paper, we present a new merging and diagnostic ontology environment called Chimaera, which was developed to address these issues in the context of HPKB. We also report on some initial tests of its effectiveness in merging tasks.

1 INTRODUCTION

Ontologies are finding broader demand and acceptance in a wide array of applications. It is now commonplace to see major web sites include taxonomies of topics including thousands of terms organized into five-level or deeper organizations as a browsing and navigation aid. In addition to broader use of ontologies, we also observe larger and more diverse staff creating and maintaining the ontologies. Companies are now hiring "chief ontologists" along with "cybrarian" staffs for designing, developing, and maintaining these ontologies. Many times the team leader may have knowledge representation or library science training, however the staff may not have much or any formal training in knowledge representation. The broader demand for ontologies along with greater diversity of the ontology designers is creating demand for ontology tools.

The average ontology on the web is also changing. Early ontologies, many of which were used for simple browsing and navigation aids such as those in Yahoo and Lycos, were taxonomies of class names. The more sophisticated ontologies were large and multi-parented. More recently, mainstream web ontologies have been gaining more structure. Arguably driven by e-commerce demands, many class terms now also have properties associated with them. Early commerce applications, such as Virtual Vineyards, included a handful of relations, and now many of the consumer electronics shopping sites are including tens or hundreds of slot names, sometimes associated with value-type constraints as well. We now see more complicated ontologies even in applications that are only using ontologies to support smart search applications. Additionally, ontologies are being used more to support reasoning tasks in areas such as configuration and intelligence tasks. A decade ago, there were a modest number of ontology-supported configurators such as PROSE/QUESTAR [McGuinness and Wright, 1998; Wright et. al., 1993], however now, web-based configurators and configurator companies such as Trilogy, Concentra, Calico, etc. are common. There are even spin offs of configurator companies handling special areas of configuration such as PC-Order for PC configuration. Configuration ontologies not only have class, slot, and value-type information, but they typically have cardinality and disjointness information that supports reasoning with contradictions. Thus, we claim that ontologies are becoming more common, the designers come from more diverse backgrounds, and ontologies are becoming larger and more complicated in their representational and reasoning needs.

Simultaneously, there appears to be a stronger emphasis on generating very large and standardized ontologies. Areas such as medicine began this task many years ago with SNOMED [Spackman, et. al., 1997] and UMLS

[McCray and Nelson, 1995]. Recently broader and shallower efforts have emerged like the joint United Nations/Dunn and Bradstreet effort to create an open coding system for classifying goods and services [UNSPSC, 1999]. Another new distributed broad ontology is the DMOZ Open Directory effort [DMOZ, 1999] with over 200,000 categories and over 21,000 registered knowledge editors. The goal of standard ontologies is to provide a highly reusable, extensible, and long-lived structure. Large ontologies in concert with the challenges of multiple ontologies, diverse staffing, and complicated representations strengthens the need for tools.

In this paper, we address two main areas. The first is merging different ontologies that may have been written by different authors for different purposes, with different assumptions, and using different vocabularies. The second is in testing and diagnosing individual or multiple ontologies. In the rest of this paper, we will give two project descriptions that served as motivation for our work on merging and diagnostic tools. We will then describe an ontology environment tool that is aimed at supporting merging and testing ontologies. We will describe the tool's used in our work on DARPA's HPKB program [Cohen, et. al., 1998]. We will also describe an evaluation of the merging capabilities of the tool. Finally, we will present the diagnostic capabilities and discuss future plans.

2 TWO MOTIVATING PROBLEMS

In the last year, some of the authors were involved in each of two major ontology generation and maintenance efforts. We gained insight from the tasks that was used to help shape our resulting ontology tool efforts. Subsequently, we have used [McGuinness, 1999] as well as licensed the tools on other academic and commercial ontology projects. We will describe the tasks briefly and present an abstraction of the problem characteristics and needs with relation to ontology tools.

2.1 MERGING THE HIGH PERFORMANCE KNOWLEDGE BASE ONTOLOGIES

The first problem was in the HPKB program. This program aimed to generate large knowledge bases quickly that would support intelligence experts in making strategic decisions. The KBs had a reasonably broad subject area including terrorist groups, general world knowledge (such as that contained in the CIA World Fact Book [Frank, et. al., 1998]), national interests, events (and their results) in the Middle East, etc. The types of questions that an analyst might ask of a KB may be simple, including straight "look up" questions like finding the leader of an organization or the population of a country. Other questions may be quite complex, including asking about the possible reaction of a terrorist group to a particular action taken by a country. Knowledge bases in this program tended to have a high degree of structure, including many slots associated with classes, value-type constraints on most slots, values on many slots, minimum cardinality constraints on slots, disjoint class information, etc. The knowledge bases were typically designed by people trained in knowledge representation and usually populated by those literate but not expert in artificial intelligence.

In the first year of the program, many individual knowledge bases were created in order to answer particular "challenge problem" questions. These questions were designed to be typical of those that a government analyst would ask. Two competitive research and integration teams were evaluated on the quality of the answers that their knowledge bases and associated reasoners returned. Many of the challenge problem questions in the first year were answered in particular contexts, i.e., with only a subset of the knowledge bases loaded. In the second year of the program, some teams, including ours, needed to be prepared to answer questions in any portion of the domain. We needed to load all of the knowledge bases simultaneously and potentially reason across all of them. Thus, we needed to load a significant number of KBs (approximately 70) that were not originally intended to be loaded together and were written by many different authors. Our initial loading and diagnosis step was largely manual with a number of ad hoc scripts. This was a result of time pressure in concert with the expectation that this was a one-time task. Some of the findings from the merging and diagnosis task were as follows:

- Large ontology development projects may require extensive systematic support for pervasive tests and changes. Our final ontology contained approximately 100,000 statements (and the version of the ontology after forward chaining rules had been run contained almost a million statements). Even though the individual ontologies all shared a common "upper ontology", there was still extensive renaming that needed to be done to allow all the ontologies to be loaded simultaneously and to be connected together properly. There were also pervasive tests to be run such as checks for comment and source field existence as well as argument order on functions. We discovered, for example, that different authors were using relational arguments in differing orders and thus type constraints were being violated across ontologies. Additionally, if a relation's domain and range constraints were used to conclude additional class membership assertions for arguments of the relation, then those arguments could end up with multiple class memberships that were incorrect. For

example, if relation Leader has a domain of Person and a range of Country, one author states "(Leader Clinton US)", and another states "(Leader US Clinton)", then Clinton could be inferred to be a person AND a country.[1]

- Repairing and merging large ontologies require a tool that focuses the attention of the editor in particular portions of the ontology that are semantically interconnected and in need of repair or further merging. There were many places where taxonomic relationships were missing when multiple ontologies were loaded together. For example, a class denoting nuclear weapons was related to the "weapon" class but not to the "weapon of mass destruction" class, nor to the disjoint partition of classes under "weapon". A tool that showed (just) the relevant portions of the taxonomies and facilitated taxonomy and partition modifications later turned out to be extremely valuable for editing purposes.
- Ontologies may require small, yet pervasive changes in order to allow them to be reused for slightly different purposes. In our HPKB task, we found a number of slots that needed to be added to classes in order to make the classes useful for additional tasks. We found many slots in the same ontology that appeared to be identical yet were unrelated. (We hypothesize that one major cause of this problem was that a term inherited a slot and value-type constraint from a parent class, but the author did not know to look for the slot under its given name, thus the author added a slot to capture the same notion under another name.) Also, we found a large number of slots that were inverses of other slots but were not related by an explicit slot inverse statement. Without the inverse information, the inverse slots were not being populated and thus were not useful for question answering even though the information appeared to be in the knowledge base. Our goal was to support users in finding the connections that needed to be made to make ontologies more useful.
- Ontologies may benefit from partition definitions and extensions. We found many ontologies that contained some disjoint partition information (e.g., "people" are disjoint from "bodies of water"), but in many cases the partition information was under specified. In the previous example with incorrect argument order, if we had information stating that people were disjoint from countries, then the inconsistency could have been detected earlier, most likely at knowledge entry time.

[1] This inference is consistent if there is no information that states that country and person are disjoint.

2.2 CREATING CLASS TAXONOMIES FROM EXISTING WEB ONTOLOGIES

In a noticeably different effort, we used a Web crawler to mine a number of web taxonomies, including Yahoo! Shopping, Lycos, Topica, Amazon, and UN/SPSC, in order to mine their taxonomy information and to build corresponding CLASSIC [Borgida et. al., 1989; Brachman, et. al., 1999] and OKBC (Open Knowledge Base Connectivity) [Chaudhri, et. al, 1998] ontologies. Our goals for this work were (1) to "generate" a number of naturally occurring taxonomies for testing that had some commercial purpose, and (2) to build a larger cohesive ontology from the "best" portions of other ontologies. ("Best" was initially determined by a marketing organization as portions of ontologies that had more usage and visibility.)

Our ontology mining, merging, and diagnosis effort had little emphasis on reasoning, but instead was centered on choosing consistent class names and generating a reasonable and extensible structure that could be used for all of the ontologies. The expected use of the output ontology was for web site organization, browsing support, and search (in a manner similar to that used in FindUR [McGuinness, 1998]).

We found that extensive renaming was required in these ontologies mined from the Web. For example, we found the unique names assumption was systematically violated within individual ontologies so that class names needed their own contexts in order to be useful. Thus, systematic treatment was required to put individual ontology branches into their own name space and to separate terms like steamers under clothing appliances from steamers under kitchen appliances. We also found extensive need for ontological reorganization. Thus, we still required focusing an editor's attention on pieces of the ontology. Additionally, we found need for more diagnostic checks with respect to ontological organization. For example, there were multiple occurrences of cycles within class graphs. So, we introduced checks for cycles into our diagnostics.

There was also no partition information in these ontologies, but there were multiple places where it appeared beneficial to add it. Our initial automatically generated ontologies were obtained from web sites that lacked explicit slot information, thus all of our slot information was systematically generated (and thus less likely to need the same kinds of modifications as those we found from human-generated slot information). Subsequent inspections of other web ontologies containing slot information, however, revealed many of the same issues we observed in our HPKB analysis work.

These two experiences, along with a few decades of staff experience with building knowledge representation and

reasoning systems and applications, led us to design and implement an ontology merging and diagnosis tool that we will describe next.

2.3 Needs Analysis

The two previous efforts motivate the following needs in a merging and diagnostic tool:

- Name searching support (across multiple ontologies)
- Support for changing names in a systematic manner
- Support for merging multiple terms into a single term
- Focus of attention support for term merging based on term names
- Focus of attention support for term merging based on the semantics of term descriptions
- Browsing support for class and slot taxonomies
- Support for modifying subsumption relationships in class and slot taxonomies
- Partition modification support
- Support for recognizing logical inconsistencies introduced by merges and edits.
- Diagnostic support for verifying, validating, and critiquing ontologies

3 AN ONTOLOGY MERGING AND DIAGNOSIS TOOL

Chimaera is a new ontology merging and diagnosis tool developed by the Stanford University Knowledge Systems Laboratory (KSL). Its initial design goal was to provide substantial assistance with the task of merging KBs produced by multiple authors in multiple settings. It later took on another goal of supporting testing and diagnosing ontologies as well. Finally, inherent in the goals of supporting merging and diagnosis are requirements for ontology browsing and editing. We will define the tasks of ontology merging and diagnosis as used in our work, and then we will introduce the tool.

We consider the task of merging two ontologies to be one of combining two or more ontologies that may use different vocabularies and may have overlapping content. The major two tasks are to (1) to coalesce two semantically identical terms from different ontologies so that they are referred to by the same name in the resulting ontology, and (2) to identify terms that should be related by subsumption, disjointness, or instance relationships and provide support for introducing those relationships. There are many auxiliary tasks inherent in these tasks, such as identifying the locations for editing, performing the edits, identifying when two terms could be identical if they had small modifications such as a further specialization on a value-type constraint, etc. We will focus on the two main tasks for our discussion.

The general task of merging can be arbitrarily difficult, requiring extensive (human) author negotiation. However, we claim that merging tools can significantly reduce both labor costs and error rates. We support that claim with the results from some initial tool evaluation tests.

We addressed the task of diagnosing single or multiple ontologies by producing a test suite that evaluates (partial) correctness and completeness of the ontologies. The major tasks involve finding and reporting provable inconsistencies, possible inconsistencies, and areas of incomplete coverage. As with merging, diagnosis can be arbitrarily complex, requiring extensive human analysis to identify all problems and present them in an order appropriate to the problem importance. Tools built to provide the first level of analysis, however, can greatly reduce human labor cost as well as improve the consistency of the analysis. In our diagnostic test suite, we do not attempt to find all problems; we just choose a subset that is computationally viable and motivated by usefulness of the reports.

3.1 CHIMAERA

Chimæra is a browser-based editing, merging, and diagnosis tool. Its design and implementation is based on our experience developing other UIs for knowledge applications such as the Ontolingua ontology development environment [Farquhar, et al, 1997], the Stanford CML editor [Iwasaki, et al, 1997], the Stanford JAVA Ontology Tool (JOT), the Intraspect knowledge server [Intraspect 1999], a few web interfaces for CLASSIC [McGuinness, et. al., 1995; Welty, 1996], and a collaborative environment for building ontologies for FindUR [McGuinness, 1998]. Chimaera has a web-based UI that is optimized for Netscape and MSIE browsers. It is written in HTML, augmented with Javascript where necessary to support niceties like spring-loaded menus and drag and drop editing.

Our goal was to make it a standards-based generic editing, merging, and diagnosis tool. When Ontolingua's editor was first developed, there was no standard API for knowledge-based systems. Since then, the OKBC API has been developed by KSL and SRI International's AI Lab. OKBC allows us to develop tools that can merge KBs in any OKBC-compliant representation system either on the same machine or over the network. Chimæra was designed from the ground up to be a pure OKBC application. Our typical editing environment is Ontolingua, but this is not a requirement. For example,

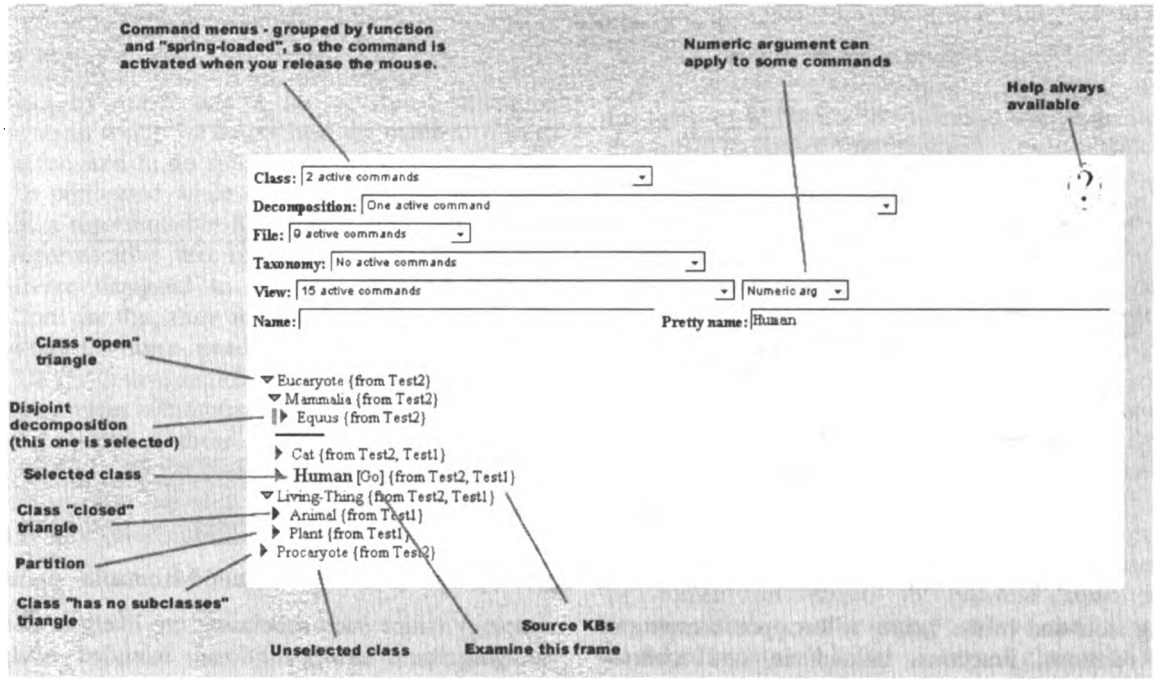


Figure 1: A view of Chimæra's user interface

one could edit in Ontosaurus {Swartout, et. al, 1996] or OntoWeb [Domingue, 1998] to produce the ontology. If the ontology editor produces OKBC-compliant files, then they can be loaded directly into Chimaera. Otherwise, indented list format, tuple format, or a few other standard forms may be used. In general, OKBC wrappers can be developed for a wide range of knowledge representation systems. For example, in one of our e-commerce ontology projects, we generated CLASSIC ontologies and developed an OKBC wrapper for CLASSIC that was used to load OKBC-compliant input into Chimaera. Translation systems such as OntoMorph [Chalupsky, 2000] could also be used to support multiple languages.

The UI for the current version of Chimæra is shown in Figure 1. The interface consists of a set of commands on spring-loaded menus (the command activates as soon as the menu selection is made). Like most GUIs, the user selects operands by clicking on them, and selection is shown by the selected operands being displayed in boldface. Applicable commands are then available on the menus, and inapplicable commands are also displayed showing the reason why they are inapplicable. The UI contains amongst its seventy odd commands a rather full-featured taxonomy and slot editor as well as commands more obviously associated with the ontology merging task, e.g., the "Merge Classes" command. It also contains 17 diagnostic commands along with options for their modification. The current UI is not a general-purpose editing environment for ontologies. It only addresses classes and slots; non-slot individuals and facets are not displayed. Similarly, there is no support for the editing of axioms. We plan to extend the functionality of the tool in later versions to include all object types. In contrast to two other merging efforts [Fridman Noy and Musen, 1999; and Chalupsky, et. al., 1997], our environment also supports creating and editing disjoint partition information and includes an extensive repertoire of diagnostic commands.

The restricted nature of the UI allows us to present a view of the KB to the user that is not cluttered by extraneous commands, widgets, or KB content. This is very important to the design of the UI, since focus of attention is vital in the KB merging task. The user may never be able to make merging decisions if the classes to be considered are many screens apart. There are, therefore (currently) no fewer than 29 different commands in the View menu that affect the way the KB is displayed to the user, and the system uses sophisticated techniques to automatically select default settings for those commands that are appropriate in most cases.

Chimaera currently addresses only a portion of the overall ontology merging and diagnosis tasks. Even though it may be viewed as an early design in terms of a complete merging and diagnostic tool, we have found significant value in it to date. We now describe some experiments designed to evaluate its usefulness in merging.

The experiments we have run only make use of those features in Chimaera designed to support the merging of class-subclass taxonomies. Chimaera includes support for

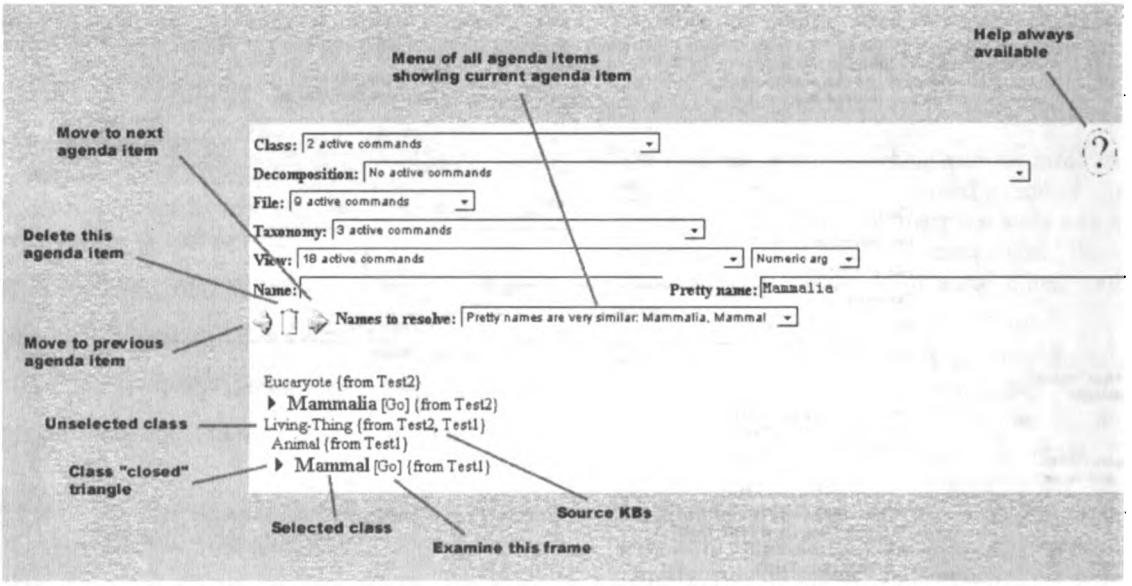


Figure 2: Chimæra in name resolution mode suggesting a merge of Mammal and Mammalia

merging slots and in the future, will support merging of facets, relations, functions, individuals, and arbitrary axioms. Similarly, the diagnosis functions only include domain independent tests that showed value in our experiments to date. These tests allow limited user input for modifications to the tests. In our future environment, we expect to include a diagnostic testing language that allows users to dynamically add new tests to the test suite, and thus support more domain-dependent diagnostics as well

3.2 MERGING AND EVALUATION

Chimaera generates name resolution lists that help the user in the merging task by suggesting terms each of which is from a different ontology that are candidates to be merged or to have taxonomic relationships not yet included in the merged ontology. For example, figure 2 shows a suggestion for merging Mammalia from ontology "Test2" with Mammal from ontology "Test1" based on the similarity of the names. The suggested candidates may be names of classes or slots. The module that puts candidates on the list is controlled by a user-settable "vigor" metric that activates a progressively more extensive search for candidate sets of terms. It considers term names, presentation names (called "pretty names" in Ontolingua), term definitions, possible acronym and expanded forms, names that appear as suffixes of other names, etc.

Chimaera also generates a taxonomy resolution list where it suggests taxonomy areas that are candidates for reorganization. It uses a number of heuristic strategies for finding such edit points for taxonomies. One looks for classes that have direct subclasses from more than one ontology (since such subclasses are likely to need some reorganization have additional intended relationships among them that are not yet in the merged ontology).

We ran four experiments aimed at evaluating Chimaera's merging effectiveness. They were focused on (1) coalescing ontology names, (2) performing taxonomic edits, (3) identifying ontology edit points, and (4) testing overall effectiveness in a substantial merging task. Because of space constraints here, we describe our high level findings and only describe one of the experiments in detail.

A long version of the merging experiment write-up is available from http://www.ksl.stanford.edu/yearindex.html.

4 EXPERIMENTAL FINDINGS

We conducted a set of experiments scoped to be within our resource budget that were designed to produce a measure of the performance of Chimæra. We also compared those results to the performance of other tools that a KB developer might reasonably use to do such a merge, absent the KB merging tool itself. At each stage in the experiment, our goal was to control for as many factors as possible and to assure that the experimental settings correspond closely to the settings in which the tool would actually be used.

KB merging is a non-trivial cognitive task, and our tools are also non-trivial, so it is not at all surprising that it should be difficult to design experiments to measure the utility of such tools. The overriding principle we used was that whenever a judgement call had to be made in the

experiment design, we tried to make sure that any bias introduced in that judgement worked *against* showing Chimæra in a good light.

We began by conducting a set of three calibration experiments in which we determined the number of steps and time required to do specific types of operations that would be performed while doing a merging task using Chimæra, a representative KB editing tool (Ontolingua), and a representative text editing tool (Emacs). These studies were designed to provide quantitative "rate" comparisons in that they indicated which steps in the merging task Chimæra speeds up and by how much, and to provide qualitative indications of the steps for which Chimæra provides substantial improvements in reliability. Using the results of these calibration experiments, we then performed a larger merge task using only Chimæra. The calibration experiments were then used to estimate the comparative utility of Chimæra over this larger task.

The primary results of these experiments are the following:

- Merging two or more substantial ontologies was essentially not doable in a time effective manner using a text-editing tool, primarily because of the difficulty of examining the taxonomy of any non-trivial ontology using only a text editor.
- Chimaera is approximately 3.46 times faster than an ontology editing tool (Ontolingua) for merging substantial taxonomies. Moreover, for the portion of the taxonomy merging task for which Chimaera's name resolution heuristics apply, Chimaera is approximately 14 times faster than an ontology editing tool (Ontolingua).
- Almost all of the operations performed during a taxonomy merge would be more error-prone if they were performed using an ontology editing tool (Ontolingua), and the critical "merge class" operations would be extremely error-prone if performed using a KB editing tool.

The ontology merging task is only an interesting problem when one tries to merge large ontologies. Chimaera has proved to provide considerable utility in non-trivial merging tasks. The other tool options tried were so poor at this task that it became impractical to perform a head-to-head experiment against other tools because the other tools simply were not able to merge reasonably large ontologies in a practical amount of time. We conclude, therefore, that Chimæra, even though it addresses only a portion of the overall merging task, makes a significant qualitative difference in one's ability to build large ontologies using fragments derived from a number of sources.

4.1 EXPERIMENT 3: FINDING EDIT POINTS

The time taken to execute an editing operation is only a minor part of the ontology merging process; a major task for the user is finding the places in the input ontologies that are to be edited. Our third experiment, which focused on that task, may be the most instructive and important; thus, we describe it in detail here.

In this experiment, we attempted to determine the relative performance of Emacs, the Ontolingua editor, and Chimæra in the edit-point finding activity. When we attempted to build scripts for these activities using Emacs, it became apparent that the task of finding good edit points is so difficult in Emacs that one simply could never realistically use Emacs for such a task. The core problem is that most of these activities involve the user being able to see the ontology's taxonomy in order to make rational decisions. It is so difficult to examine the taxonomy of any non-trivial ontology using Emacs that the user would be forced, in effect, to reinvent some sort of representation system using either shell scripts or keyboard macros in order to have a chance of knowing what to do. We decided that this was sufficiently unrealistic that we eliminated Emacs from Experiment 3.

The idea behind Experiment 3, therefore, was to try to measure the time taken by a user to find candidate edit points using the Ontolingua editor and Chimæra. Our goal in designing the experiment was to factor out the time actually taken to perform the suggested edits, since that editing time was considered in Experiment 2.

Our goal was to measure the performance of users performing the edit-point-finding task in as unbiased manner as possible. In the best of all possible worlds, we would have a large pool of input ontologies and of test subjects with the necessary skills so that we could get an overall idea of the performance of the tools. This was not practical, so we selected a pair of test subjects who were as closely matched as we could find in knowledge representation skill as well as skill in the use of the tool. We used one subject with the Ontolingua editor who had considerable experience using the tool as a browser, though little as an editor. The test subject who was to use Chimæra had a small amount of experience using Chimæra as a browser, but no experience using any of the editing features. In accordance with our overall strategy of bias reduction, the bias in the test subject selection clearly favored the Ontolingua editor over Chimæra.

The test subject using Chimæra was given a guided tour of its editing operations. Both users had about two hours to practice using their respective tools specifically on the ontology merging task. For the practice session, they were each provided with some small sample ontologies that had no overlap with the ontology content of the test ontologies.

We instrumented Chimæra so that we could identify the commands being invoked, the times at which the commands were executed, and the number of arguments used by the commands. The goal was for the test subject to use Chimæra not only to find the edit points, but also to perform the edits so that we could learn as much as possible from the process. Having performed the timed merge, we would then subtract out the time taken to perform the edits to make the results more comparable to the use of the Ontolingua editor.

For the experiment with the Ontolingua editor, we timed the test subject with a stopwatch. When the test subject identified an edit point, the clock was stopped, and the test subject turned away from the screen. The desired edits were then performed, and the clock restarted. It is essential to perform the edits suggested because performing the proposed edits changes the structure of the ontology and the way that it appears on the display. This often results in terms that were previously distant on the display appearing close together, thereby making other candidate edit operations more obvious. Ironically, as we saw in our previous calibration experiments, Chimæra is so much more efficient at performing these edits than the Ontolingua editor that when the experiment referee stepped in to perform the requested edits during the experiment, he used Chimæra to perform them.

We decided that before conducting the experiment it would be a good idea to calibrate the experiment by getting an idea of the upper bound of the possible performance using Chimæra. We therefore had one of the developers of Chimæra - and experienced user - use Chimæra to perform the merge of the two proposed input ontologies a number of times. This was to give us an idea both of the number of edit operations that could reasonably be found in the ontologies, and the maximum speed with which a user could perform the merge if all of the thinking time necessary to decide what to merge was reduced as close to zero as possible. We anticipated that were we to graph the edit operations against time we would see a clear knee in the curve at the point at which the "low-hanging fruit" had all been plucked. Given this point, we intended to run the real experiment for a time not exceeding the time for the knee in the curve. This would, we thought, give us some confidence that we were likely to be in the low-hanging fruit operating region of both tools, since we anticipated that Chimæra would be faster than the Ontolingua editor. If we were to stop the experiment after an arbitrary time without performing this calibration, we might bias the result in favor of the slower tool if the faster tool had been fast enough to get outside its low-hanging fruit region in the time allotted, but the slower tool had not.

Figure 3 shows the results from this calibration experiment. The anticipated knee in the curve did not actually appear, though there is a knee at 622 seconds, where the user finished his first pass through the Name Resolution agenda. After about an hour, the user stopped, reporting that all of the edits that he was performing seemed by that point to be concentrated in cleaning up one of the input ontologies, rather than actually merging the ontologies. Given the results of this calibration run, we decided to give the two test subjects 55 minutes in which to perform their edit-point finding.

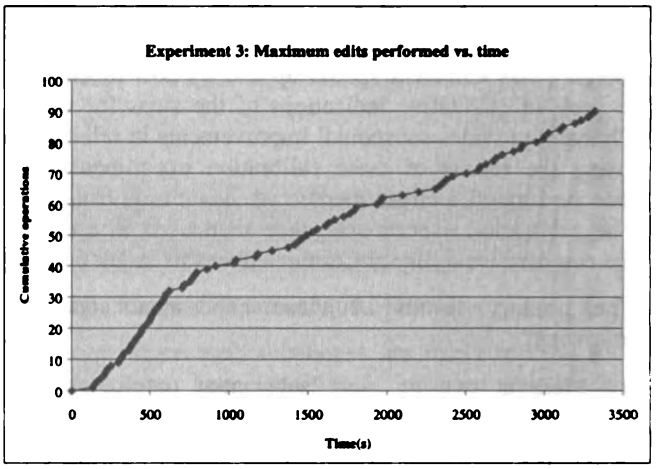


Figure 3: The cumulative number of edit points found and edits performed using Chimæra plotted against time in seconds. The dense set of points early on was characterized by a large number of merge operations driven from the Name-Resolution agenda. Subsequent edit points were found using the Taxonomy-Traversal agenda and other Chimæra browsing features.

We wanted to try to control as much as possible for the different representational decisions that two test subjects might make, so during the experiment, the developer who performed the calibration experiment was on-hand to answer any representational questions whenever such questions arose. This oracle was not allowed to volunteer any edit suggestions, but would, if prompted, say whether any pair of classes should be merged or have an additional relationship asserted.

Our goal was to define the idea of "finding an edit point" to mean as closely as possible the same thing for the two tools. There were, however, some differences because of the nature of the tools. For Chimæra, the time taken to find an edit point was the time taken to identify the place to edit and to select the arguments for the editing operation. These times were captured by Chimæra's instrumentation. For the Ontolingua editor, we measured the time taken between the clock being started and the user calling out for the clock to be stopped upon finding a candidate edit and calling out the arguments to be used in the edit. We did this because we knew that we would not

be using the Ontolingua editor actually to perform the edits.

We had to decide whether to allow edits within input ontologies as well as across ontology boundaries during the experiment. We anticipate that the ontology merging process may typically involve some clean up to the input ontologies, but we wanted to focus our evaluation on the merging process rather than on intra-ontology editing, so we decided to disallow intra-ontology edits for this experiment. Cycorp provided to us for use in this experiment the part of their IKB relating to Agents that they had built for the HPKB program. With this input ontology, which we had not seen before, we could be confident that we had received a clean and well-documented ontology, and that there would be a reasonable amount of overlap with our own Agents representation similarly developed for the HPKB program. Restricting scoring of candidate edit points to include only proposed edits that had at least one argument from each ontology was easy in Chimæra, since they are color-coded. In the case of the Ontolingua editor, we renamed all of the terms in one ontology to have a common suffix indicating the ontology of origin. We provided this substantial help to the subject using Ontolingua in order to improve the comparability of the experimental results. Interestingly, we used a command in Chimæra to do this systematic renaming.

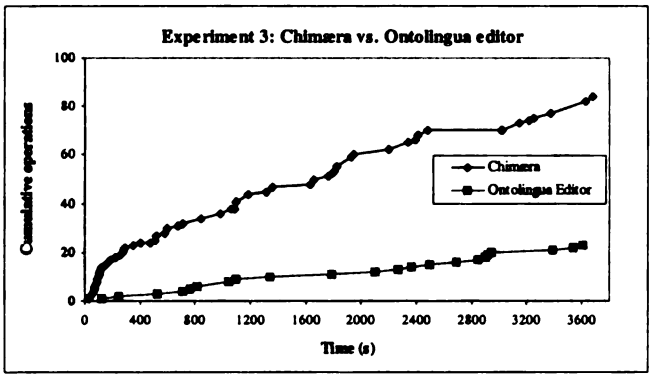


Figure 4: Chimæra proved to be superior to the Ontolingua editor at finding candidate edit points

Figure 4 shows the results from Experiment 3. There are a number of ways to interpret at these results depending on what we hope to learn. The fact that the curve for Chimæra is always well above the curve for the Ontolingua editor clearly shows the significant superiority of Chimæra for this task. Overall, the Chimæra test subject was able to identify 3.7 times as many edit points, and was also able to perform the edits.

It is difficult to come up with a simple numeric rate for the number of edit points found per minute because we need to have reason to believe that there is an underlying linear model before such a rate number has any meaning or predictive value. Luckily, we do have reason to believe that there is a linear model underlying at least part of Experiment 3 (there may be linear models underlying other parts, but we do not know this with certainty). The Name Resolution menu in Chimæra presents the user with a simple list of candidate edit points. The user, in general, iterates through this list until all elements of the list have been processed. This is what our user was doing during the first 185 seconds of the experiment. In this linear region, merges were being found, considered, and accepted or rejected at the rate of about one every nine seconds. The correlation to a linear fit in this region is R^2=0.94, supporting our reasoning that the underlying model is linear. This result is consistent with our experience on other ontologies. We expect that this linear model should be broadly independent of ontology size, since the time taken to construct the agenda itself is factored out. The algorithm that constructs the name resolution menu is $O(n^2)$. Within the known linear region, Chimæra outperformed the Ontolingua editor by a factor of fourteen.

The test subject who was using the Ontolingua editor also, we believe, exhibited a linear strategy. This happened because during the practice session the test subject developed a systematic method for finding candidate edit points that involved a systematic traversal of the ontology. The strategy involved looking in turn at each class and then considering all of its siblings and the siblings of each of its superclasses up to the roots to see whether a merge or taxonomic edit was appropriate. The number of examination steps necessary for any given class is a function of the depth of the taxonomy and branching factor of the superclasses. The time taken to perform any given iteration in this strategy is therefore reasonably constant, though influenced by the number of subclasses that must be inspected once a candidate edit has been selected. We believe that the complexity of this strategy is bounded by $O(n.\log_b(n))$ and $O(n^2)$, where b is the average subclass branching factor and n is the size of the ontology. For an ontology such as the one we used as input, however, the model should be roughly linear. The results for a linear fit throughout the experiment for the Ontolingua editor give us a rate of 157 seconds per edit point found, with a correlation of 0.98.

5 DIAGNOSTICS TESTS

Chimaera also has a set of diagnostics that can be run selectively or in their entirety. Routinely when we obtain knowledge bases now, we run the diagnostics and invariably find issues with our incoming KBs. The current list of diagnostics was derived as a retrospective analysis of the most useful domain independent tests that we needed to run on the HPKB and on the crawled web ontologies. They group into four areas:

1) Simple checks for incompleteness (missing argument names, missing documentation strings, missing sources, missing type constraints, missing term definitions);

2) Syntactic analysis (incidence of words (or sub-strings), possible acronym expansion);

3) Taxonomic analysis (redundant super classes, redundant types, trivial instances or subclasses of THING, definition extensions from included ontologies), and

4) Semantic evaluation (slot value/type mismatch, class definition cycle, domain/range mismatch).

This is obviously not everything that could be checked. The current diagnostic suite does not connect to the full theorem prover so there is only limited consistency checking. The current testing environment also does not give users the power to add their own, potentially domain-specific, checks. Even with the limited power of the diagnostics set though, we successfully used it to provide initial correctness and completeness checks of all incoming HPKB knowledge bases for our final team evaluation. Possibly more importantly, its output was usable by people with little training in knowledge representation, and we found that with no training they could make effective and correct improvements to the knowledge bases guided by the diagnostic output. Also, the tool takes multiple input formats, thus we were able to allow people to use it who had no familiarity with OKBC or Ontolingua. We had some SNARK and KIF-literate users load in their ontologies in the input format they were familiar with, run diagnostics, and debug their knowledge bases with little intervention from us. We also used this toolset to check for problems in our semi-automatically generated ontologies from web crawls. The tests found a surprising number of things that would have been tedious or difficult for us to find ourselves, such as class cycles and inconsistency in naming in Amazon's ontology. Finally, we used the merging tool with ontologies generated by naïve users with no training, and they were able to immediately merge independent ontologies and use the tool effectively to focus their attention on the problem areas in the ontologies. They also used the diagnostics effectively with no training.

6 CONCLUSION

We have presented an ontology editing, merging, and diagnostic environment developed to meet the emerging needs of representation and reasoning tasks on the Web and of ontology creation and maintenance tasks. We have briefly overviewed the merging and diagnostics components and presented some evaluation results on the merging side and some anecdotal reports on the diagnostics side. While our tool is in its early stages, these evaluations of the tool, our own personal use of the tool, and demand for the tool from both the commercial and academic sectors provide notable evidence that it makes significant improvements in productivity and quality of ontology development and maintenance. We are continuing to develop the tool, focusing in particular on extending its reasoning capabilities, semantic analysis, its extensibility, and usability by non-experts.

Acknowledgements

The authors are indebted to DARPA for their support of this research under contract N66001-97-C-8554, *Large-Scale Repositories of Highly Expressive Knowledge*. We would also like to thank Cycorp for kindly providing knowledge base fragments for some of the merging experiments and Bob Schrag at IET for his support in the design and conducting of the experiments.

Bibliography

Alex Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick; *CLASSIC: A Structural Data Model for Objects*, Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, Portland, Oregon, June, 1989, pp. 59--67.

Ronald J. Brachman, Alex Borgida, Deborah L. McGuinness, and Peter F. Patel-Schneider; *"Reducing" CLASSIC to Practice: Knowledge Representation Theory Meets Reality*; In the Artificial Intelligence Journal, 114(1-2) pages 203-237, October, 1999.

Hans Chalupsky, Eduard Hovy, Tom Russ; *Progress on an Automatic Ontology Alignment Methodology*; Proceedings of the ANSI ad hoc group on ontology standards, 1997. http://ksl-web.stanford.edu/onto-std/ .

Hans Chalupsky. OntoMorph: A Translation System for Symbolic Knowledge, in A. G. Cohn, F. Giunchiglia, and B. Selman, editors, Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000), Morgan Kaufmann Publishers, San Francisco, CA., 2000.

Vinay Chaudhri, Adam Farquhar, Richard Fikes, Peter Karp, and James Rice; *OKBC: A Programmatic Foundation for Knowledge Base Interoperability*; Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98); AAAI Press. (http://www.ksl.stanford.edu/KSL_Abstracts/KSL-98-08.html)

Paul R. Cohen, Robert Schrag, Eric Jones, Adam Pease, Albert Lin, Barbara Starr, David Easter, David Gunning, and Murray Burke. The DARPA High Performance

Knowledge Bases Project. In Artificial Intelligence Magazine. Vol. 19, No. 4, pp.25-49, 1998.

DMOZ – Open Directory Project. http://www.dmoz.org .

John Domingue. *Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web.* 11th Knowledge Acquisition for Knowledge-Based Systems Workshop, April 18th-23rd. Banff, Canada, 1998.

Adam Farquhar, Richard Fikes, and James Rice; *The Ontolingua Server: a Tool for Collaborative Ontology Construction*; International Journal of Human-Computer Studies **46**, 707-727, 1997. (http://www.ksl.stanford.edu/KSL_Abstracts/KSL-96-26.html)

Gleb Frank, Adam Farquhar, and Richard Fikes; *Building a Large Knowledge Base from a Structured Source: The CIA World Fact Book*; IEEE Intelligent Systems, Vol. 14, No. 1, January/February 1999. (http://www.ksl.stanford.edu/KSL_Abstracts/KSL-98-16.html)

Natalya Fridman Noy and Mark A. Musen. *An Algorithm for Merging and Aligning Ontologies: Automation and Tool Support* in AAAI-99 Workshop on Ontology Management. Also, SMI Technical Report SMI-99-0799.

Natalya Fridman Noy and Mark A. Musen; *SMART: Automated Support for Ontology Merging and Alignment*; Twelfth Banff Workshop on Knowledge Acquisition, Modeling, and Management; Banff, Alberta, Canada; 1999. Also, SMI Technical Report SMI-1999-0813

Intraspect Knowledge Server, Intraspect Corporation, 1999. (http://www.intraspect.com/product_info_solution.htm)

Y. Iwasaki, A. Farquhar, R. Fikes, & J. Rice; *A Web-based Compositional Modeling System for Sharing of Physical Knowledge*. Morgan Kaufmann, Nagoya, Japan, 1997. (http://www.ksl.stanford.edu/KSL_Abstracts/KSL-98-17.html).

Kevin Knight and Steve Luk; *Building a Large-Scale Knowledge Base for Machine Translation*; Proceedings of the National Conference on Artificial Intelligence (AAAI), 1994.

A.T. McCray and S.J. Nelson; *The representation of meaning in the UMLS*; Methods of Information in Medicine; 1995; 34: 193-201.

Deborah L. McGuinness; *Ontologies for Electronic Commerce*, Proceedings of the AAAI Artificial Intelligence for Electronic Commerce Workshop, Orlando, Florida, July, 1999.

Deborah L. McGuinness; *Ontological Issues for Knowledge-Enhanced Search*; Proceedings of Formal Ontology in Information Systems, June 1998. Also in Frontiers in Artificial Intelligence and Applications, IOS-Press, Washington, DC, 1998.

Deborah L. McGuinness and Peter Patel-Schneider; *Usability Issues in Knowledge Representation Systems*; Proceedings of the Fifteenth National Conference on Artificial Intelligence, Madison, Wisconsin, July, 1998. This is an updated version of *Usability Issues in Description Logic Systems* published in Proceedings of International Workshop on Description Logics, Gif sur Yvette, (Paris) France, September, 1997.

Deborah L. McGuinness, Lori Alperin Resnick, and Charles Isbell; *Description Logic in Practice: A CLASSIC: Application*; Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal, Canada; August, 1995.

Deborah L. McGuinness and Jon Wright; *An Industrial Strength Description Logic-based Configurator Platform*; IEEE Intelligent Systems, Vol. 13, No. 4, July/August 1998, pp. 69-77.

Deborah L. McGuinness and Jon Wright; *Conceptual Modeling for Configuration: A Description Logic-based Approach*; Artificial Intelligence for Engineering Design, Analysis, and Manufacturing Journal - special issue on Configuration, 1998.

K.A. Spackman, K.E. Campbell, and R.A. Cote; *SNOMED-RT: a reference terminology for health care*; Proceedings of the 1997 AMIA Annual Fall Symposium; 1997; 640-644.

William Swartout, Ramesh Patil, Kevin Knight, and Tom Russ; *Toward Distributed Use of Large-Scale Ontologies*, Proceedings of the 10th Banff Knowledge Acquisition Workshop; Banff, Alberta, Canada; November 9-14, 1996.

United Nations Standard Product and Services Classification (UNSPSC) Code organization. http://www.unspsc.org/home.htm

Chris Welty; *An HTML Interface for CLASSIC*; Proceedings of the 1996 International Workshop on Description Logics; AAAI Press; November, 1996.

Long-Term Maintainability of Deployed Knowledge Representation Systems

Nestor Rychtyckyj
Manufacturing Quality Business Systems
Ford Motor Company
Dearborn, MI 48121
nrychtyc@ford.com

Robert G. Reynolds
Department of Computer Science
Wayne State University
Detroit, MI 48202
reynolds@cs.wayne.edu

Abstract

The use of knowledge representation systems for building large complex ontologies to support real-world problems has been demonstrated in various application areas. One such system is Ford's Direct Labor Management System (DLMS) that has been used since 1990 in the very dynamic domain of process planning for vehicle assembly. The long-term maintenance of the DLMS knowledge base has demonstrated both the flexibility and reliability of semantic network-based approaches in a rapidly changing industrial setting. This maintainability becomes very difficult over time due to changes in all of the following areas: the external business environment, the processes and physical concepts being modeled, and the underlying hardware and software architecture.

The requirement for maintaining the DLMS knowledge base over the last ten years has given us a unique perspective into the various maintenance problems and issues that need to be addressed. This paper will discuss those issues and our approaches utilized in maintaining a KR system over a prolonged period of time. We will also discuss our facility for automatic knowledge base validation and the use of an evolutionary computational technique known as Cultural Algorithms for knowledge base re-engineering.

1 INTRODUCTION

The use of KL-ONE and associated knowledge representation systems for building large complex knowledge bases to support real-world problems has been demonstrated in various application areas (Brachman et al 1991). One such system is Ford's Direct Labor Management System (DLMS) that has been used since 1990 in the very dynamic domain of process planning for vehicle assembly (O'Brien et al 1989, Rychtyckyj 1999). The long-term maintenance of the DLMS knowledge base has demonstrated both the flexibility and reliability of semantic network-based knowledge bases in a rapidly changing industrial setting. The most critical issue in utilizing knowledge-based systems over a long period of time is the maintainability of the system. This maintainability can become very difficult due to changes in all of the following areas: the external business environment, the processes and physical performance and produce undesirable results.

Having maintained the DLMS knowledge base over the concepts that are being represented and, in the hardware and software architecture that the system must utilize. All of these factors introduce modifications to the system that can cause unforeseen problems that impact the system last ten years has given us a unique perspective into the various problems and issues that need to be addressed. These issues focus on the processes and tools that are needed to validate and verify the knowledge base since it is updated frequently. This allows us to keep up with rapidly changing market conditions. The modifications made to a semantic-network based knowledge base will also impact the structure and design

of the network, and may degrade the system performance over time if adjustments are not made.

One of our goals in writing this paper is to improve the communication between the KR research community and the business world where knowledge-based systems are deployed and maintained. Our paper will focus on the maintainability of our knowledge base and discuss several new approaches to improve maintenance. One such approach is a method to automatically generate test cases to validate the correctness of the knowledge base as part of the maintenance process. Another approach is use evolutionary computation to analyze the knowledge base as a tool to help us maintain and re-engineer it as required. We will also give specific examples of knowledge base design decisions that had either a positive or negative affect on future maintenance. With this paper we hope to demonstrate what issues are important in maintaining a knowledge base in a dynamic business environment over a long-term period.

In this paper we will discuss the design of the KL-ONE-based DLMS knowledge base and it's use in the domain of automobile assembly planning. A brief description of DLMS is contained in Section 2. Section 3 will focus on the validation and verification of the DLMS knowledge base, and discuss the various tools that were developed for this task. The issues in maintaining the DLMS knowledge base in production over a ten-year period are examined in Section 4. Section 5 details the use of an Evolutionary Computational technique, known as Cultural Algorithms, as a tool in re-engineering the DLMS knowledge base.

2 THE DIRECT LABOR MANAGEMENT SYSTEM

The Direct Labor Management System (DLMS) is utilized by Ford Motor Company's Vehicle Operations division to manage the use of labor on the assembly lines throughout Ford's vehicle assembly plants. DLMS was designed to improve the assembly process planning activity at Ford by achieving standardization within the vehicle process build description and to provide a tool for accurately estimating the labor time required to perform the actual vehicle assembly. In addition, DLMS provides a framework for allocating the required work among various operators at the plant. It also builds a foundation for the automated machine translation of the process descriptions into foreign languages, a necessity in the current global business market.

The standard process-planning document, known as a process sheet, is the primary vehicle for conveying the assembly information from the initial process planning activity to the assembly plant. A process sheet contains the detailed instructions needed to build a portion of a vehicle. A single vehicle may require thousands of process sheets to describe its assembly. The process sheet is written by an engineer utilizing a restricted subset of English known as Standard Language. Standard Language allows an engineer to write clear and concise assembly instructions that are machine-readable.

This process sheet is written by an engineer at the Vehicle Operations General Office; it is then sent to the DLMS system to be "validated" before it can be released to the assembly plants. Validation includes the following: checking the process sheet for errors, generating the sequence of steps that a worker at the assembly plant must perform in order to accomplish this task and calculating the length of time that this task will require. The DLMS system interprets these instructions and generates a list of detailed actions that are required to implement these instructions at the assembly plant level. These work instructions, known as "allocatable elements", are associated with MODAPTS (MODular Arrangement of Predetermined Time Standards) codes that are used to calculate the time required to perform these actions. MODAPTS codes are widely utilized within Industrial Engineering as a means of measuring the body movements that are required to perform a physical action and have been accepted as a valid work measurement system. (IES 1988)

The allocatable elements generated by DLMS are used by engineering personnel at the assembly plant to allocate the required work among the available personnel. DLMS is a powerful tool because it provides timely information about the amount of direct labor that is required to assemble each vehicle, as well as pointing out inefficiencies in the assembly process.

The DLMS system consists of five main subsystems: parser, analyzer, simulator, knowledge base manager, and the error checker. The input into DLMS is a process sheet. This is initially parsed in order to break down the sentences into their lexical components that include the verb, subject, modifiers, prepositional phrases and other parts of speech. Since Standard Language is a restricted subset of English, the parser has a very high rate of success in properly parsing the input from the process sheets. The parser utilizes the Augmented Transition Network (ATN) method of parsing. The analyzer then uses the components of the parsed element to search the knowledge base (or taxonomy) for relevant information describing that item. For example, if the input element contained the term "BUMPER", the taxonomy will be searched for the term "BUMPER". If it is found, the system will acquire all of the attributes that "BUMPER" has: (it is a Part, its size is large, it needs mechanical assistance to be moved, etc.) The system performs this analysis on all of the components of the parsed input

element in order to select what work instructions are required. Any process element that is not parsed successfully will then be flagged by one of the error rules. These rules will (hopefully) suggest to the user how to correct this element. The work instructions are then found in the taxonomy based on all of the available input and are passed to the simulator.

The simulator uses the information found in the taxonomy to generate the allocatable elements and MODAPTS codes that describe the input elements. These work instructions are then sent to the user. The knowledge base manager is used to maintain the knowledge base. The system administrators based on input provided from the Industrial Engineering community perform this maintenance process.

All of the associated knowledge about Standard Language, tools, parts, and everything else associated with the automobile assembly process, is contained in the DLMS knowledge base or taxonomy. This knowledge base structure is derived from the KL-ONE family of semantic network structures, and is an integral component in the success of DLMS. DLMS also contains a rulebase of over 350 rules that are used to drive the validation process and perform error checking on the Standard Language input.

The organization of the knowledge base is based on the KL-ONE model. The root of the semantic network is a concept known as THING that encompasses everything within the DLMS world. The children of the root concept describe various major classes of knowledge, and include such concepts as TOOLS, PARTS and OPERATIONS. Each concept contains attributes or slots that describe that object. The values of these attributes are inherited from the concept's parents. Ranges of valid values can be given for any particular attribute. Any attempt to put an invalid value in that attribute will trigger an error. All of the information dealing with the organization and structure of the taxonomy is also contained in the taxonomy itself. There are four types of links that describe the relationship between any two concepts: subsumes, specializes, immediately-subsumes and immediately-specializes. The subsumption relation describes a link between a parent concept and all of its children, including descendants of its children. The "immediately-subsumes" relation describes only the concepts that are direct descendants of the parent concept. The "specializes" and "immediately specializes" relations are inverses of the subsumption relation. A concept "immediately specializes" its direct parent concepts and "specializes" all of the concepts that are ancestors of its parents. These relationships are stored as attributes of any given concept. They can be utilized to trace any concept through the entire taxonomy.

The DLMS system utilizes a classification algorithm to create concepts and place them into their appropriate position in the taxonomy. The classifier utilizes various attributes of the concept in order to place it into its correct position. These "classifiable" attributes are slot values that play a major role in determining where a concept belongs. For example, the attribute "size" is very important in classification, while the "output format" slot has little value in classification. Classification is performed by finding the appropriate subsumers, linking the concept in and then locating all the concepts that should be subsumed by the new concept. The system narrows this search procedure considerably by selecting the appropriate node in the concept to begin the classification process. The concept that is to be classified is placed at the starting node; the system then tries to push the new concept node as far down the tree as possible. The classifiable attributes are used as objective measures to determine if the concept is in its proper place. Within DLMS this classification algorithm is applied to all of the instances of the input string that describe the process element. In a simple element this may include a verb, an object and an associated tool. When the classifier is complete, each of the above instances will inherit necessary values from the knowledge base in order to build the appropriate operation to describe the required actions. Figure 1 illustrates a diagram of a simple classification procedure.

Currently the DLMS taxonomy contains over 9500 concepts and each concept may contain up to 69 different attributes or properties. These concepts are divided into more than 800 classes and the number of links within the system exceeds 110,000. The knowledge encoded into our knowledge base can be divided into Standard Language lexical terms (1/3 of the knowledge base) with the remainder being the tools, parts and operations that describe our manufacturing assembly process. Each concept is described in terms of its properties and its links to parent and child nodes. . Figure 2 gives the definition of the concept of "HAMMER" as it appears in the DLMS taxonomy.

The following concept description displays the attributes and links that exist for the concept of HAMMER. This term is used to describe HAMMER as both a verb and a noun. Therefore it contains attributes that are utilized when HAMMER is a verb (linear-dimension-formula) and when HAMMER is used as a tool (tool-type). The decision of how to use HAMMER is made in the parser when it reads in the initial input sentence. For example, the sentence "Hammer the hammer" is correctly processed and the appropriate actions are generated.

(THING HAMMER ((ATTRIBUTE LINEAR-DIMENSION-FORMULA (0.0 . 1.0) (:USER ("NESTOR" 3025977316))) (ATTRIBUTE TOOL-TYPE COMMERCIAL-TYPE (:INHERITANCE (COMMERCIAL-TOOL))) (ATTRIBUTE VERB-REQUIRED-CASES (OP-OBJECT) (:USER ("BRICE" 2797675436))) (ATTRIBUTE VERB-OPTIONAL-CASES (OP-TOOL) (:USER ("BRICE" 2797675417))) (ATTRIBUTE POS (NOUN VERB) (:USER ("BRICE" 2793212943))) (ATTRIBUTE SIZE SMALL-SIZE (:INHERITANCE (SMALL-OBJECT))) (ATTRIBUTE IMMEDIATELY-SPECIALIZES (STRIKE-PROCESS-TYPE-VERB HAND-TOOL) ((:USER ("WOODHEAD" 2820233809)) (:USER ("BRICE" 2793212933)))) (ATTRIBUTE SPECIALIZES (NON-PRODUCT-TOOL TOOL-PROCESS-TYPE-VERB PROCESS-TYPE-VERB STRIKE-PROCESS-TYPE-VERB VERB STANDARD-LANGUAGE-VERB COMMERCIAL-TOOL OBJECT TOOL GENERIC-TOOL SMALL-OBJECT HAND-TOOL LEXICAL-NODES THING) ((:INTERNAL) (:INTERNAL) (:INTERNAL) (:USER ("WOODHEAD" 2820233809)) (:INTERNAL) (:USER ("BRICE" 2797671095)) (:INTERNAL) (:INTERNAL) (:INTERNAL) (:INTERNAL) (:INTERNAL) (:USER ("BRICE" 2793212933)) (:INTERNAL) (:INTERNAL))) (ATTRIBUTE TYPE PRIMITIVE (:USER ("SCOTT" 2786960833)))))

Figure 2: Concept Description for Term HAMMER

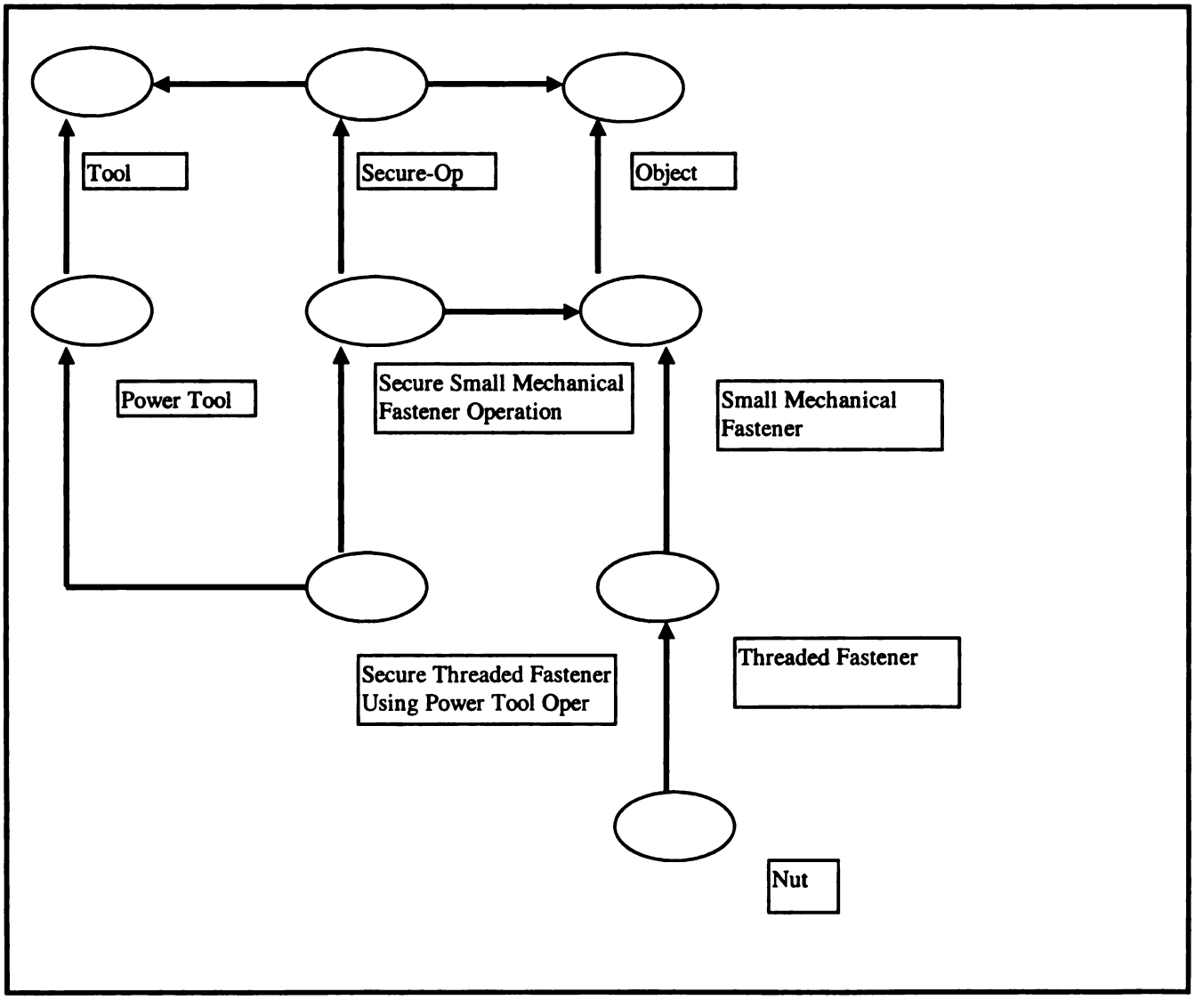


Figure 1: Classification Diagram in DLMS

Another type of concept that is utilized in the DLMS taxonomy is that of an operation. An operation describes the specific physical actions that assembly personnel must do in order to fulfill a particular instruction. These actions may describe other actions, which must also be retrieved in order to completely describe this operation. For example, the attribute "actions" contains a description that refers to the WALK operation. During processing the classifier will retrieve and utilize the appropriate actions for walking as part of the Obtain operation. The attributes entitled "subsumes", "specializes", "immediately-subsumes" and "immediately specializes" describe the structural links that this concept has to other concepts in the semantic network. Figure 3 displays the concept description for the OBTAIN-SMALL-OBJECT operation.

(THING OBTAIN-SMALL-OBJECT-OP ((ATTRIBUTE MODAPTS M4G3 (:USER ("NESTOR" 3058725286))) (ATTRIBUTE IMMEDIATELY-SUBSUMES (OBTAIN-TACK-CLOTH-OP OBTAIN-TAPE-OP) ((:USER ("NESTOR" 3011004008)) (:USER ("NESTOR" 2923308896)))) (ATTRIBUTE SUBSUMES (OBTAIN-TACK-CLOTH-OP OBTAIN-TAPE-OP) ((:USER ("NESTOR" 3011004008)) (:USER ("NESTOR" 2923308896)))) (ATTRIBUTE ACTIONS (((OP-VERB WALK) (|to| *OP-OBJECT*)) *MODAPTS* ((OP-VERB WALK) (|to| UNIT))) (:USER ("VITALE" 2867670890))) (ATTRIBUTE OBJECT-QTY ONE (:INHERITANCE (OBTAIN-OBJECT-OP))) (ATTRIBUTE OP-VERB OBTAIN (:INHERITANCE (OBTAIN-OBJECT-OP))) (ATTRIBUTE DOCUMENTATION "G. Miko - 11/07/88" (:USER ("WOODHEAD" 2804171544))) (ATTRIBUTE OP-OBJECT SMALL-OBJECT (:USER ("WOODHEAD" 2804171461))) (ATTRIBUTE TYPE GENERIC (:USER ("WOODHEAD" 2804171457))) (ATTRIBUTE OP-VEHICLE VEHICLE (:INHERITANCE (OPERATION))) (ATTRIBUTE OP-PLANT AGO (:INHERITANCE (OPERATION))) (ATTRIBUTE PROCESS-PREFIX PREFIX-TYPE (:INHERITANCE (OPERATION))) (ATTRIBUTE IMMEDIATELY-SPECIALIZES (OBTAIN-OBJECT-OP) ((:USER ("WOODHEAD" 2804171450)))) (ATTRIBUTE SPECIALIZES (OBTAIN-OP OBTAIN-TYPE-OPERATION OBTAIN-OBJECTS-OP THING OPERATION OBTAIN-OBJECT-OP) ((:INTERNAL) (:INTERNAL) (:INTERNAL) (:INTERNAL) (:INTERNAL) (:USER ("WOODHEAD" 2804171450))))))

Figure 3: Concept Description of the Obtain-Small-Object Operation

3 VALIDATION AND VERIFICATION OF THE DLMS KNOWLEDGE BASE

As mentioned previously, the DLMS taxonomy or knowledge base contains all of the relevant information that describes the vehicle assembly process at Ford Motor Company. This includes all of the lexical classes included in Standard Language such as verbs, nouns, prepositions, conjunctions and other parts of speech, various tools and parts utilized at the assembly plants, and descriptions of operations that are performed to build the vehicle.

The DLMS Knowledge Base is maintained through the use of the Knowledge Base Manager (KBM). The Knowledge Base Manager is a graphical tool that is used by the system developers to make important changes to the knowledge base that will affect the actual output generated by the system. Since this output will have a major impact on the assembly process, any such change must be approved by a committee representing all of the interested parties. All changes made to the knowledge base are logged by the system in order to keep a record of the system's modification history.

DLMS is currently integrated into two separate Ford process-planning systems. The original mainframe-based system is still being utilized at some of our plants in North America. A new client-server based process planning system has been developed and is currently being rolled out to our assembly plants.

The Knowledge Base Update (KBU) is an automated update facility that was used by system users to make modifications to the knowledge base. It is still utilized by the users of the old mainframe process planning system, but has not been implemented in the new global client server system due to maintenance issues that will be discussed in the next section.

The first order of business in knowledge base validation and verification is to develop a baseline of test results that have been manually inspected and judged to be correct by the user community. Any changes to the knowledge base are migrated into production only after the regression tests have been completed and accepted. A utility is used to run the series of regression tests to test the system output for a variety of inputs and the results are then compared to the previous baseline. Any changes that have been introduced are manually examined and the change is either accepted or rejected. If the change is accepted, a new baseline will be created for this test. A rejected test forces the developers to correct the knowledge base until the regression test is deemed to be acceptable.

The suite of regression tests must be constantly updated to add and to remove tests as the situation warrants. This updating of the regression tests is done in two different ways: manually and automatically. Manual updates are done when a developer recognizes that a certain script must be added in order to check a particular scenario. These manual scripts are usually complex, and they include a series of tests that have caused problems for the users. We have also developed a utility that generates test cases automatically against the knowledge base. This is done by reading through the knowledge base, generating a script that will test all of the properties of a particular node by analyzing the attributes of that node and creating a test that will check those attributes. Each of these test cases is then executed; the results are then compared against the previous baseline. As with the manually created text cases, the base line is updated to reflect any changes that have been made to the knowledge base.

The automatic test case generation utility works on the class of "Operations" that describe the actions that are needed to accomplish a particular task. The system reads through all of the concepts in this class, selects the appropriate attribute values and creates a Standard Language sentence that would test all of these attributes. The file containing all of these sentences is then used as input into the DLMS regression tests. The results from these sentences are compared against the baseline and any discrepancies are flagged for manual inspection. Figure 4 shows a sample of the test cases that are automatically generated.

```
(APPLY DAUB)
(APPLY TWO DAUBS)
(APPLY FLUID TO 10 INCH LONG X 12 INCH WIDE
AREA)
((APPLY 0.125 INCH DIAMETER X 12 INCH LONG
SEALER BEAD TO OBJECT USING GUN) (TOOL 1 1 C
COMM 10 GUN))
((APPLY 0.5 INCH THICK X 0.5 INCH WIDE X 10
INCHES LONG SEALER RIBBON TO OBJECT USING
GUN) (TOOL 1 1 C COMM 10 GUN))
((APPLY LARGE DAUB USING GUN) (TOOL 1 1 C
COMM 10 GUN))
((APPLY BEAD USING GUN) (TOOL 1 1 C COMM 10
GUN))
((APPLY RIBBON SEALER USING RIBBON-SEALER-
GUN) (TOOL 1 1 C COMM 10 GUN))
((APPLY SEALER USING GUN) (TOOL 1 1 C COMM 10
GUN))
```

Figure 4: Sample Test Cases

There are many advantages to utilizing tools for automatically creating test cases. These tools provide a quick and easy method to generate a complete set of test cases for a production knowledge base. Each test case is generated directly from the knowledge base and provides an up-to-date snapshot of the current state of the knowledge base. This utility also provides us the capability to create a report describing the knowledge base that is disseminated to the user community.

However, the use of automatic generation tools has not replaced all of the manually created test scripts due to the following reasons. First, the test creation utility cannot generate complex test scripts that may require a sequence of instructions to fully represent a particular scenario. Second, since the knowledge base is frequently changed, it is often necessary to modify the test utility to keep it current and complete. Third, it is not possible to use the test utility to generate any scripts that model missing or invalid knowledge since that knowledge is not already contained in the knowledge base. Nevertheless, we have found the use of automated test utilities to be a very useful and productive method to assist with knowledge base maintenance.

4 MAINTENANCE ISSUES IN DLMS

As mentioned previously, the Knowledge Base Update (KBU) facility was designed to allow engineering personnel to update the DLMS knowledge base directly without having to contact the developers. The KBU consists of an update screen interface on the mainframe system that provides a facility for adding, updating, or querying the knowledge base. The mainframe system generates a request that is then sent to the DLMS system for processing. The users are notified if their request has been implemented or rejected due to an error in the way the request was created.

Our experience indicates that user editing of the knowledge base has not been very successful either from the user viewpoint or from the developer side. The editing of the knowledge base requires a deeper understanding of the knowledge representation scheme than is needed for updating a spreadsheet or database. This necessitated the creation of a complex user interface to the KBU that many users found difficult to master. Also, most of the user changes to the system consisted of lexical information, which required properties, such as part of speech to be specified. This was often done incorrectly and introduced errors into the system. This meant that the developers had to spend time reviewing and correcting user edits in order to catch these types of errors. Other problems were caused by users adding misspelled terms, alternate spellings, and different abbreviations for the same terminology. Previously these types of errors did not usually affect the results of

accessing the knowledge base, and were often not modified unless they were viewed to potentially cause a processing problem.

This situation changed radically with a recent corporate reorganization that required our process planning system to be used at our European plants. This added a new requirement that our process sheets be translated into the native language of our assembly plants. Due to the restricted nature of Standard Language, and the large amounts of data needed for translation, we decided to utilize Machine Translation as the means for doing these translations. We quickly discovered that the inclusion of various slang terms, misspellings, and alternate spellings were causing a huge problem for our Machine Translation system. It became evident that Ford technical terminology also had different meanings in various parts of the company. Our solution to help alleviate these problems was to centralize the updating of the knowledge base within the development group in North America under the direction a user committee, the Standard Language Committee. This Standard Language committee has representatives from the entire user community and must approve all changes to the knowledge base. Currently problems with misspellings and alternate spellings are handled by using a synonym attribute that allows for the same concept to be described in alternate ways. We utilize both internal and industry glossaries in order to check on these alternate spellings and reject those that are not known to our engineering community. This process is manually intensive, but is very critical for business that operates in a global environment.

A knowledge representation system must be very flexible in order to survive over a prolonged period of time in a dynamic environment. Rule bases have been shown to be very difficult to maintain and in many cases had to be completely rewritten so as to function in a production environment (Soloway 1987). Our experience has shown that semantic network based knowledge bases are much more flexible and can be easily adapted to handle frequent changes. The DLMS knowledge base was not originally designed to handle manufacturing processes for electric cars or flexible fuel vehicles but was easily adapted for those processes. For example, the requirements for changing the Ford business to include Jaguar was not originally conceived but caused few problems in implementation.

This flexibility is due to both the structure of the knowledge representation scheme and the software in which is implemented. Our semantic network also contains information about the internal representation of the network that allows for easy modification of that structure. For instance, new attributes along with their domain and range values can be easily be added into the structure using the graphical interface tool.

Our knowledge base was implemented initially, and currently remains, in the LISP language. The LISP language environment is an excellent framework for both creating and modifying complex representation systems. The DLMS system itself has been ported from a LISP machine platform to a UNIX platform, but the underlying LISP software has remained in production. We utilize the LISP Works tool from Harlequin, which gives us direct access to the ORACLE database as well as providing functionality for integration to other programming modules into our system.

5 RE-ENGINEERING THE DLMS KNOWLEDGE BASE USING CULTURAL ALGORITHMS

Over the lifetime of the DLMS knowledge base there have been thousands of updates that have been made to reflect modifications in the Ford manufacturing process-planning domain. The structure of the knowledge base has been radically changed as a result of these updates. Many of the underlying assumptions about network structures have been made over time, and it is not always clear that these assumptions are still valid. We are also concerned over the response time of the system as the knowledge base is growing and under constant modification. All of these factors contributed to our need to develop a method to analyze and re-engineer the DLMS knowledge base.

Our maintenance history over the last ten years shows that approximately 10 - 15% of our knowledge base is updated annually. These statistics represent the number of attributes that are modified during the given year. This number varies from year to year based on external business factors related to the deployment of the system. The location and vehicle that is being assembled will greatly influence the amount of knowledge base changes that are needed. For example, launching the system for a brand new vehicle at an overseas assembly plant will require much many more knowledge base updates than launching the system for an existing vehicle at a North American plant.

In the early years of development the number of knowledge base updates were well above 50%, but this number has stabilized as the system has matured. We expect these numbers to rise somewhat as we begin to integrate more overseas car programs into the system.

Our numbers may also appear somewhat lower than other systems (Soloway 1987) because our system contains a fairly large amount of "inactive knowledge". This describes processes and concepts that have been added into the knowledge base and not yet been implemented for various business reasons. However these concepts are included in the total size of the knowledge base and affect the knowledge base maintenance metrics.

One specific model of evolutionary computation, known as Cultural Algorithms (Reynolds 1996), has been successfully used to re-engineer a commercial rule based expert system (Sternberg and Reynolds 1997), utilized for software testing (Ostrowski and Reynolds 1999), and as a knowledge discovery system utilizing decision trees (Al-Shehri 1997). We decided to apply the Cultural Algorithm approach to the DLMS system because of the dynamic nature of our knowledge base. These updates increase the complexity of the system, the cost of maintenance, and the time required to make the necessary changes. The algorithms used to determine subsumption can be NP hard (Woods 1991). Consequently, the execution time of the system increases exponentially with the complexity of the semantic network. All of these factors have motivated us to explore the use of Cultural Algorithms as a tool to learn how to re-engineer the DLMS semantic network, and to provide Ford with a means of reducing the complexity of the semantic network structure while keeping up with the changes inherent in the current global marketplace.

Cultural Algorithms are dual inheritance systems that utilize a population and a belief space. The population consists of individuals, here representing attributes, which are modified via evolutionary operators such as mutation. The belief space contains problem-specific knowledge, which is used to guide the evolutionary process. The belief space and population communicate through a protocol that determines which members of the population can update the belief space. Figure 3 shows the basic CA components.

Knowledge base re-engineering is accomplished by using Cultural Algorithms to evolve the most efficient graphical structure that preserves the knowledge found in the original version. The system utilizes the attributes that are stored within the nodes to learn a "defining set" of attributes that can then be used to re-engineer various portions of the network. This approach provides a tool to assist in knowledge base maintenance, to improve the efficiency of the retrieval algorithms, and to reduce the complexity of the existing network. Another goal of our work is to reduce the complexity of the search process by identifying nodes that are closer to the leaf nodes and starting the search process at that point.

Initially we created a utility that analyzed the entire knowledge base by ranking all of the classes in the network in terms of *class homogeneity* based on their attribute usage. The class homogeneity is a value that represents the relationship between the members in the class and the values for the attributes that they use. Classes with high homogeneity contain categories of concepts that are very similar to each other based on their attributes. Conversely, classes with lower homogeneity contain members that that are more dissimilar in terms of attribute usage. The class homogeneity ranges from 1.0 to 0.02 for the class containing the entire network here.

Class Homogeneity is determined by the following formula:

Class Homogeneity = (number of attributes utilized by members in class) / (number of members in class * count of attributes available)

In other words, each class has a set of potential attributes that may be utilized by all of the members in that class. We analyze each member of that class and count the number of attributes that are actually being used. This total is then divided by the number of potential attributes to derive the homogeneity value for that class. A class that has a homogeneity value of 1.0 describes a class where every member in the class utilizes every attribute of that class. Classes with low homogeneity generally include members who have considerable differences between them and consequently utilize different attributes to describe their properties. As you traverse the network from the leaf nodes to the root the class homogeneity value decreases. The class homogeneity level for the entire network is a minuscule 0.02 which signifies that the entire network contains very dissimilar members. After ranking all of the classes in the network we then selected classes with different homogeneity rankings and performed a Cultural Algorithm analysis on each class. This was accomplished using the following strategy. The goal of the Cultural Algorithm was to evolve a semantic network for a given class that would be correct, and would minimize the number of defining attributes needed. This would reduce the cost of subsumption based upon a set of training instances. These results were then compared with the "defining attributes" identified by human developers over the life of the DLMS system for the same classes. Table 1 summarizes these results.

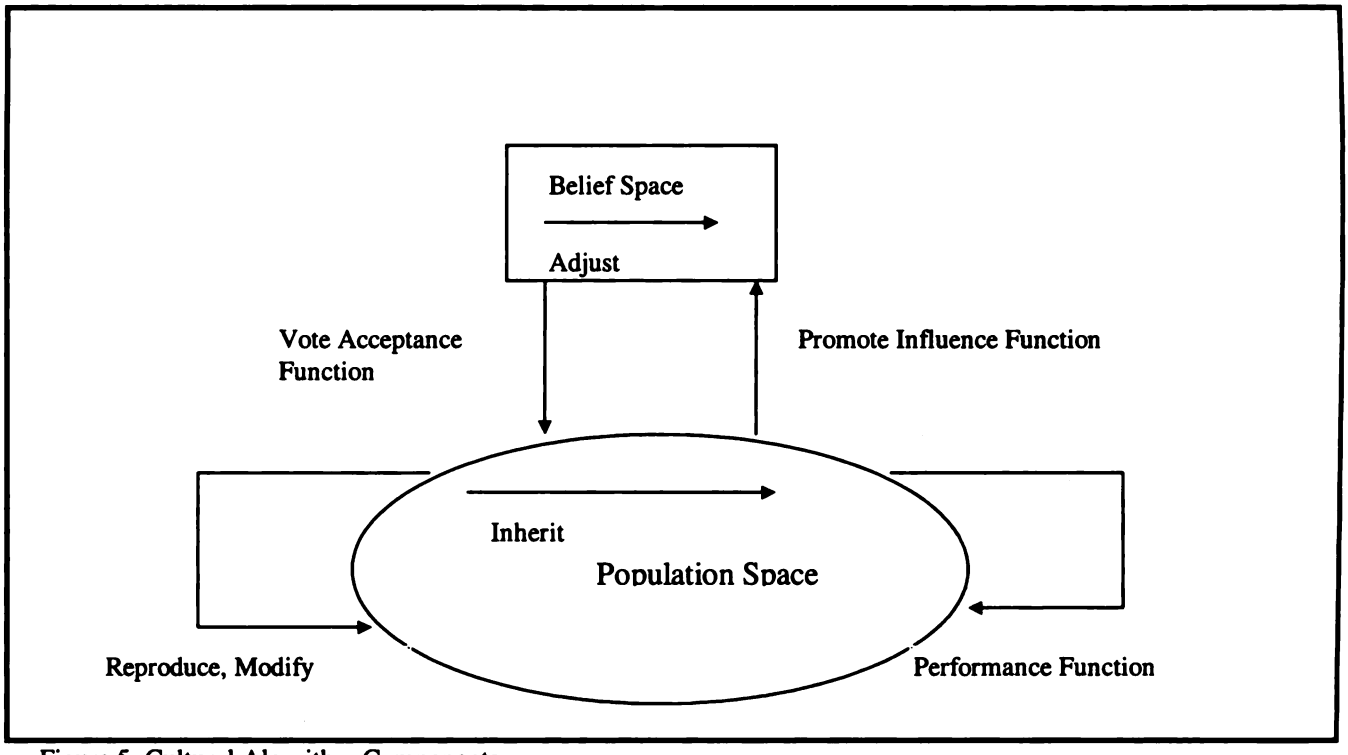


Figure 5: Cultural Algorithm Components

Class Name	Number of Attributes	Number of Members	Class Homogeneity	Defining Attributes Using CA	Defining Attributes (Manual)
Vehicle-area	5	49	1	2	N/A
Pull-Operations	10	66	0.97	2	7
Obtain-op	11	38	0.82	8	7
Position-Operations	14	28	0.67	8	9
Power-tools	9	32	0.56	4	3
Small-mechanical-fastener	9	70	0.49	3	2
Taxonomy-Relation	14	87	0.45	10	N/A
Commercial-tools	13	150	0.32	3	4
Manufacturing-Aid	13	26	0.29	3	4
Standard-Language Verbs	18	235	0.2	4	3

Table 1: Cultural Algorithm Analysis Results

Table 1 shows a representative sample of the classes in our knowledge base sorted in order of class homogeneity. The columns describe, in order, the class names, number of attributes that can be utilized by members of that class, the number of concepts contained in the class, the class homogeneity value, the number of defining attributes identified by the Cultural Algorithm system and the number of defining attributes identified by the developers manually. The first interesting result deals with the two classes that have a N/A in the Manual Attributes class. This shows that these two classes are not being actively utilized within the system during the classification process. Further investigation showed that one of the classes was developed, but never implemented due to business reasons. The second class is only used to represent structural knowledge about the knowledge base and is used directly during classification.

A comparison between the actual counts of defining attributes generated by the human designers and the CA system shows that the Cultural Algorithm approach is very close to the manual results in almost all cases. The biggest difference (2 attributes using CA's / 7 attributes using manual) occurs in a class with very high homogeneity. In this type of class, every attribute is used in almost every instance of the class. Thus, many subsets of attributes would produce the same results. The Cultural Algorithm approach detected this, and used a heuristic to select only a small subset of the attributes to be the defining ones. Another interesting result shows that the Cultural Algorithms perform very well with classes of both high and low homogeneity. This fact is very encouraging because the cost associated with finding defining attributes utilizing Cultural Algorithms is a fraction of the time that is required for developers.

This utilization of Cultural Algorithms provides us with a tool to analyze the semantic network automatically, and provides a vehicle for re-engineering the knowledge base in order to improve process efficiency and reduce network complexity. This method will also be used to analyze data before it is added into the knowledge base in order to determine how best to utilize the attributes contained in the data for maximal subsumption efficiency.

Our future plans include expanding the use of Cultural Algorithms to other aspects of the re-engineering process. The use of evolutionary computational techniques often leads to results that were not anticipated by the developers and provides a valuable tool that is useful for both maintaining and building a knowledge base.

6 CONCLUSIONS

In this paper we discussed the issues relevant to long-term utilization of knowledge representation systems based on our experience at Ford with DLMS. These included the use of validation and verification tools, processes and tools for updating the knowledge base, and the use of evolutionary computation to assist in re-engineering the knowledge base. Our experience has shown that knowledge representation systems based on semantic networks provide an excellent framework for developing systems that can exist in a dynamic environment. Long-term maintenance of such systems requires both the development of processes to support the system, as well as the corresponding software tools needed to implement these processes. Flexibility is the key requirement of knowledge representation systems as the business environment will certainly change in ways that could not be anticipated by the developers. It is also important to utilize new approaches as they become available in order to assist the developers in maintaining and re-engineering the knowledge base over its life cycle. All of these factors contribute to the successful use of knowledge representation systems in very dynamic problem domains as evidenced here.

ACKNOWLEDGEMENTS

The Direct Labor Management System is a product of the work of many people that have contributed to the success of the system over the years; therefore we would like to give credit to the following people: John O'Brien, Tom Kaszamarek, Scott Hatfield, Wayne Johnson, Richard Woodhead, Alan Turski, Henry Brice, Tom Vitale, Jay Zaback, Rick Keller and Michael Rosen.

We would also like to thank the anonymous referees for their many constructive criticisms, which helped us to improve the paper in many ways.

REFERENCES

Brachman, R., Schmolze, J., (1985), "An Overview of the KL-ONE Knowledge Representation System," *Cognitive Science* 9(2), pp. 171-216.

Brachman, R., McGuiness, D., Patel-Schneider, P., Resnick, L., Borgida, A., (1991) "Living With Classic: When and How to Use a KL-ONE-Like Language" in Principles of Semantic Networks, ed. J. Sowa, pp. 401-456, Morgan Kaufmann Publishers.

Hasan, A., (1997), "Evolution-Based Decision Tree Optimization Using Cultural Algorithms," Ph.D. Dissertation, Wayne State University.

Industrial Engineering Services (1988), Modapts Study Notes for Certified Practitioner Training.

O'Brien, J., Brice, H., Hatfield, S., Johnson, W., Woodhead, R., (1989), "The Ford Motor Company Direct Labor Management System," in *Innovative Applications of Artificial_Intelligence*, ed. Schorr & Rappaport, MIT Press, pp. 331-346.

Ostrowski, D., Reynolds, R.G. (1999), "Knowledge-Based Software Testing Agent Using Evolutionary Learning with Cultural Algorithms" in *Proceedings of the 1999 Congress of Evolutionary Computation*, Washington D.C, July 6-9, vol. 3, pp. 1657-1663, IEEE Press.

Reynolds, R.G., Chung, C. (1996), "A Self-adaptive Approach to Representation Shifts in Cultural Algorithms" in *Proceedings of the 1996 IEEE International Conference on Evolutionary Computing*, Nagoya Japan, IEEE Press, pp. 94-99.

Reynolds, R.G., Chung, C. (1997), "Regulating the Amount of Information Used for Self-Adaptation in Cultural Algorithms," in *Proceedings of the Seventh International Conference on Genetic Algorithms*, pg. 401-408, Morgan Kaufmann Publishers.

Rychtyckyj, N. (1996), "DLMS: An Evaluation of KL-ONE in the Automobile Industry". in *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning*, 588-596. Morgan Kaufmann Publishers.

Rychtyckyj, N., Reynolds, R.G., (1998), "Learning to Re-Engineer Semantic Networks Using Cultural Algorithms" in *Evolutionary Programming VII*, Springer-Verlag, pg. 181-190.

Rychtyckyj, N., (1999), "DLMS: Ten Years of AI for Vehicle Assembly Process Planning", *AAAI-99/IAAI-99 Proceedings*, Orlando, FL, July 18-22, 1999, pp. 821-828, AAAI Press.

Rychtyckyj, N., and Reynolds, R., (1999), "Using Cultural Algorithms to Improve Performance in Semantic Networks", *Proceedings of the 1999 Congress of Evolutionary Computation*, Washington D.C, July 6-9, vol. 3, pp. 1651-1656, IEEE Press.

Soloway, E., Bechant, J., Jensen, K., (1987), "Assesing the Maintainability of XCON in RIME: Coping with the Problems of a Very Large Rule-Base" in *Validating and Verifying Knowledge-Based Systems*, pp. 294-299, IEEE Press.

Sternberg, M., Reynolds, R. (1997), "Using Cultural Algorithms to Support Re-Engineering of Rule-Based Expert Systems in Dynamic Performance Environments: A Case Study in Fraud Detection" in *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 4, pp. 225-243.

Woods, W. A. (1991), "Understanding Subsumption and Taxonomy" in *Principles of Semantic Networks*, ed. J. Sowa, pp. 45-94. Morgan Kaufmann Publishers.

Revision: an application in the framework of GIS

Eric Würbel*, Robert Jeansoulin†, Odile Papini‡

Abstract

Geographical information systems (GIS) use incomplete and/or incertain information, which can be inconsistent and requires the definition of revision operations. Since GIS use a large amount of data, defining effective revision operations requires a suitable adjustment of existing strategies. In this paper, we propose to apply a revision strategy, based on inconsistency minimization, to a real geographical problem. We use a logical knowledge representation, based on propositional calculus, in order to represent the problem by sets of clauses. We generalize and characterize the notion of "removed set", that is the smallest set of clauses to drop out in order to restore consistency. In order to compute the removed sets, we adapt the Reiter's algorithm for diagnostics, and we propose several heuristics which allow to significantly reduce the time and space complexity. We then provide experimental results which show that the approach gives good results.

1 Introduction

Geographic information systems deal with incomplete and uncertain information. Since the items of information come from different sources characterized by various degrees of confidence, these items can conflict and require a revision operation. When a new item of information is added to a knowledge base, inconsistency can result, revision means modifying the knowledge base in order to maintain consistency, while keeping the new information and removing the least possible previous information.

Knowledge bases revision is both an old and important problem arising in knowledge representation, in artificial intelligence, and a lot of work has been developped since 1980. Among logical approaches, the AGM paradigm [1, 3], in which revision is interpreted as beliefs change, has become a standard. However most of the proposed approaches have been developped at the theoretical point of view and unless some exceptions [20, 19, 17, 9, 3, 13], few approaches have been implemented and few applications have been studied [24]. In other respects, most of the approaches, particularly semantic approaches, are characterized by a high complexity [9] which makes then untractable for large size problems.

In the context of geographic information systems, we deal with problems with a large amount of data and an effective implementation of revision operations requires a suitable adjustment of existing strategies.

In this paper, we propose the application of a revision strategy, based on inconsistency minimization, to the study developped by the CEMAGREF [1] about the flooding of the Herault river (France) in order to provide a better understanding of flooding phenomenon [21]. The area is segmented into compartments, a compartment is a spatial entity in which the water height is considered to be constant. The first source of information consists in aerial pictures which show hydraulic relations between some neighbouring compartments, this information is naturally incomplete. The observation of the emergence of some objects (vineyard cultures, dykes,...), conflated with the cadastral and ordnance ground knowledge, provides the second

*LIM. CMI, Université de Provence 39 rue Joliot-Curie, 13453 Marseille cedex 13, France.

†LIM. CMI, Université de Provence 39 rue Joliot-Curie, 13453 Marseille cedex 13, France.

‡GECT, Université de Toulon et du Var, BP132, 83957 La Garde cedex, France.

[1] French research center on environmental problems, more specifically concerning rivers and forests.

source of information. It allows to determine, for some compartments, a first estimation of minimal and/or maximal submersion heights, this information is incomplete and moreover inaccurate. These two sources of information can conflict and since the knowledge about the hydraulic flows is more reliable, we revise the information about submersion heights by the information about the hydraulic relations.

We propose a logical representation of the problem in the framework of propositional calculus. The estimations of submersion heights are represented by a set of positive monoliteral clauses, denoted by E. The hydraulic relations between compartments and the description of the domain of submersion heights are represented by a set of binary negative clauses and of n-ary positive clauses, denoted by I. The revision strategy stems from the inconsistency minimization of the set $E \cup I$. We propose a method of determination of the smallest subsets of clauses to remove in order to restore consistency, called kernels, which generalize the notion of removed sets developped in [20]. In order to compute the kernels, we adapt the Reiter's algorithm used to the determine conflicts, in the framework of diagnosis [22, 23].

The paper is organized as follows. In section 2 we propose a revision strategy based on inconsistency minimization. A removed sets generalization and a removed sets characterization by means of kernels is first presented, we then show how we adapt the Reiter's algorithm in order to compute the kernels. Section 3 presents the application of the revision strategy to the flooding problem. We present in section 4 the experimental results and we compare them to those previously obtained using other resolution methods, binary decision diagrams and flexible CSP before concluding in section 5.

2 Revision strategy

We now present our revision strategy based on inconsistency minimization.

2.1 Context and notations

We work in the framework of propositional calculus. As usual, a clause is a disjunction of literals. A set of clauses is considered as the conjunction of the clauses contained in the set. Let C be a finite set of clauses. We denote the collection of inconsistent subsets of C by $\mathcal{I}(C)$. We denote the collection of minimal-inconsistent subsets of C by $\mathcal{M}(C)$.

Let I and E be two finite sets of clauses, defined as follow :

- $I = I_d \cup I_c$ and $I_d \cap I_c = \emptyset$.
- I_d contains two types of clauses : positive n-ary clauses, that is $(d_1 \vee \ldots \vee d_n)$ and negative binary clauses $\neg d_i \vee \neg d_j$, $i \in \{1, \ldots, n\}$, $i \neq j$.
- I_c containts binary negative clauses.
- E contains monoliteral positive clauses.

This representation corresponds to a CSP representation in the propositional calculus as presented in [8].

Let's suppose that I is consistent and that E is consistent too, but that $I \cup E$ is inconsistent. Our goal is to define a revision operation $E*I$ to determine which initial assessments we have to drop out in order to restore consistency. We now describe the revision strategy, using the notion of kernel, and we propose heuristics based on the syntactical structure of our problem in order to get an efficient algorithm.

2.2 Removed sets and kernels

In [20], Papini defines a revision operation for a set of clauses, allowing the revision of such a set by *one* clause. This operator uses the notion of *removed set*, defined as follows:

Definition 2.1. *A removed set R for the revision operation $E*I$ is the smallest subset of clauses to remove from E such that $(E \cup I) \setminus R$ is consistent.*

The Papini's revision operation satisfies the first five AGM postulates [20]. Before going further on, we recall the notion of minimal-inconsistent subset of clauses. A finite set of clauses C is minimal-inconsistent iff C is inconsistent and $\forall S$, $S \subset C$, S is consistent. Two consequences of this definition are of special interest for the characterization of removed sets.

Consequence 2.1. *If R is a removed set for the revision operation $E * I$, then for all $c_i \in R$, $(I \cup (E \setminus R)) \models \neg c_i$.*

The proof is simple and is conducted by supposing that $(I \cup (E \setminus R)) \not\models \neg c_i$.

Consequence 2.2. *$R \subseteq E$ is a remove set for the revision operation $E*I$ iff R is a minimal set (according to cardinality) such that $(I \cup (E \setminus R))$ is consistent.*

The proof of ($\Rightarrow$) is conducted by observing that $(I \cup (E \setminus R))$ is consistent and proving its maximal consistency. The ($\Leftarrow$) way is proved by showing that

$(I \cup (E \setminus R)) \cup \{\neg c_i \mid c_i \in R\}$ is consistent and that R is the minimal set verifying this property.

In [22], Reiter uses the notion of hitting set of a collection, which intersects with each element of the collection. We call *kernel* a minimal hitting set and we show that removed sets can be represented by kernels.

Definition 2.2 (Hitting set and kernel). *Let F be a collection of sets. A* hitting set *of F is a set $H \subseteq \bigcup_{S \in F}$ such that $\forall S \in F$, $H \cap S \neq \emptyset$.*

H is a kernel *(or minimal hitting set) of F iff H is a hitting set of F and $\forall H' \subset H$ with $H' \neq \emptyset$, H' is not a hitting set of F.*

We denote by $\mathcal{N}(F)$ the collection of the kernels of collection F. We show that a kernel from $\mathcal{N}(\mathcal{I}(I \cup E))$ is a removed set characterization.

Theorem 2.1. *$R \subseteq E$ is a removed set for the revision operation $E * I$ iff R is a kernel of $\mathcal{I}(I \cup E)$, the collection of inconsistent subsets of $(I \cup E)$.*

The proof of ($\Rightarrow$) is quite obvious. The proof of ($\Leftarrow$) is developped in two steps. First we show that $(I \cup (E \setminus R))$ is consistent and then we show that R is the minimal set (in term of cardinality) verifying this property by showing that $\forall c \in R$, $[I \cup ((E \setminus R) \cup \{c\})]$ is inconsistent.

Starting with the characterization theorem 2.1, we propose the following revision strategy:

1. determine the remove sets by the computation of $\mathcal{N}(\mathcal{I}(I \cup E))$, using the algorithm described in following sections,
2. define an order upon these removed sets in order to choose which one to pick to restore consistency.

2.3 Initial algorithm description

Let $\mathcal{F}$ be a collection of sets. We want to compute the collection $\mathcal{N}(\mathcal{F})$ of kernels of $\mathcal{F}$. Reiter proposed an algorithm to compute kernels in the framework of diagnosis [22]. This algorithm contained some errors, and it has been corrected by [23]. We first present the corrected version of the algorithm from an informal point of view. The algorithm can compute $\mathcal{N}(\mathcal{F})$ without extensively knowing $\mathcal{F}$. Basically, it is a hitting sets computing algorithm to which a set of rules is added in order to ensure the minimality of computed hitting sets. It uses a *breadth-first* n-ary tree building. Each tree node is labeled by an element of $\mathcal{F}$, or by "$\surd$" , which means that we have exhausted $\mathcal{F}$.

Level 0: The root is labeled by an element $F_0 \in \mathcal{F}$. If $\mathcal{F}$ is empty, the root is labeled by "$\surd$" (in this case the algorithm stops : there are no kernels).

Level 1: From the root, we build as many branches as elements in the labeling set F_0. Each branch is labeled by the corresponding element in F_0. For each branch labeled by f_0^i, we produce a new node. This node is labeled by an element $F_1^i \in \mathcal{F}$ such that $f_0^i \notin F_1^i$. If there is no such F_1^i, the new node is labeled by "$\surd$".

Level 2: From each level 1 node i labeled by F_1^i, we build as many branches as elements in F_1^i. Each branch is labeled by the corresponding element in F_1^i. For each branch labeled by f_1^i , we generate a new node, which is labeled by an element $F_2^j \in \mathcal{F}$ such that $\left\{f_0^i, f_1^j\right\} \not\subseteq F_2^j$. If there is no such F_2^j, the new node is labeled by "$\surd$".

And so on until all branches end with a node labeled by "$\surd$". More formally:

Definition 2.3 (HS-tree). *Let $\mathcal{F}$ be a collection of sets, a tree T is an HS-tree iff it is the smallest tree having the following properties:*

1. *Its root is labeled by an element from $\mathcal{F}$. If $\mathcal{F}$ is empty, its root is labeled by "$\surd$".*
2. *If n is a node from T, let $H(n)$ be the set of branch labels on the path going from the root of T to n. If n is labeled by "$\surd$", it has no successor in T. If n is labeled by a set $F \in \mathcal{F}$, then, for each $f \in F$, n has a successor node n_f in T, joined to n by a branch labeled by f. The label of n_f is a set $F' \in \mathcal{F}$ such that $F' \cap H(n_f) = \emptyset$, if such a set exists. Otherwise, n_f is labeled by "$\surd$".*

Some immediate properties of HS-trees [22, 23]: (a) If n is a node labeled by "$\surd$", then $H(n)$ is a hitting set of $\mathcal{F}$. (b) For each kernel $F \in \mathcal{F}$, there exists $n \in T$, n being labeled by "$\surd$", such that $H(n) = F$.

From these two properties, Reiter, and Grenier, Smith and Wilkerson later on, develop techniques allowing to compute only kernels of $\mathcal{F}$ and to avoid unecessary accesses to $\mathcal{F}$. Note that corrections added by [23] transform the tree into a directed acyclic graph. We now sum up these techniques:

1. Tree generation is made in breadth first order : this avoids the generation of nodes n such that there exists a node n' from the same level with $H(n) = H(n')$.

2. Node reuse: If a node n labeled with a set $F \in \mathcal{F}$, and if n' is a node such that $H(n') \cap F \neq \emptyset$, n' is labeled with F. By this, we avoid an unnecessary access to $\mathcal{F}$.

3. Tree pruning:

   (a) If a node n is labeled by "$\surd$" and there exists a node n' such that $H(n) \subseteq H(n')$, close node n', do not calculate its successors.

   (b) If we are about to generate a node n and there exists a node n' such that $H(n) = H(n')$, make the branch leading to n' point to n[2].

   (c) If nodes n and n' have been labeled respectively by F and F' and $F' \subset F$, then for each $f \in F \setminus F'$, delete branches starting from F and labeled by f, until you reach a node having more than one ancestor[3].

Example: Let us consider the following collection $\mathcal{F} = \{\{a,b\},\{b,c\},\{a,c\},\{b,d\},\{b\}\}$. We comment the construction of the HS-tree shown in figure 1. The

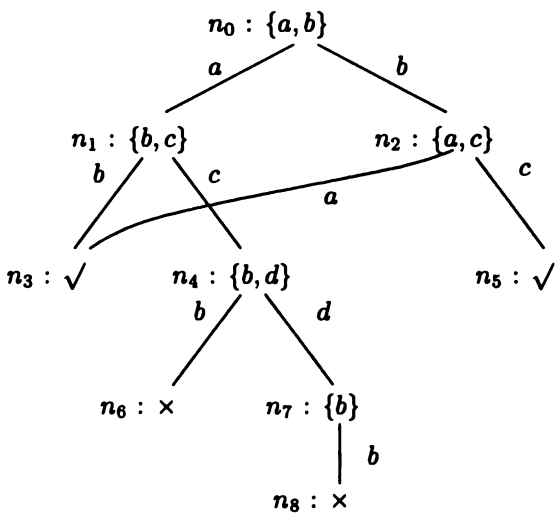


Figure 1: Example of an HS-Tree

tree is built in a breadth-first order as follows:

n_0 Access to $\mathcal{F}$ and pick up a set $F \in \mathcal{F}$. In this case, $F = \{a,b\}$. Build two branches labeled by elements of F.

n_1 There are no reusable node, so we pick up a new set $F \in \mathcal{F}$ such that $\forall f \in F$, $f \notin H(n_1) = \{a\}$ and we label n_1 by F. We build two branches labeled by elements of $\{b,c\}$.

n_2 There are no reusable node, so we pick up a new set $F \in \mathcal{F}$ such that $\forall f \in F$, $f \notin H(n_2) = \{b\}$ and we label n_2 by F. The branch starting from n_2 and labeled by a verifies prunning rule 3b, so we join it with branch b starting from n_1.

n_3 There is no set $F \in \mathcal{F}$ such that $\forall f \in F$, $f \notin H(n_3) = \{a,b\}$, so we label the node by "$\surd$".

n_4 There are no reusable node, so we pick up a new set $F \in \mathcal{F}$ such that $\forall f \in F$, $f \notin H(n_4) = \{a,c\}$ and we label n_2 by F. We build two branches labeled by elements of $\{b,d\}$.

n_5 There is no set $F \in \mathcal{F}$ such that $\forall f \in F$, $f \notin H(n_5) = \{b,c\}$, so we label the node by "$\surd$".

n_6 The node n_3 is labeled by "$\surd$" and is such that $H(n_3) \subset H(n_6)$, so we close the node, according to rule 3a.

n_7 There are no reusable node, so we pick up a new $F \in \mathcal{F}$ such that $\forall f \in F$, $f \notin H(n_4) = \{a,c\}$ and we label n_2 by F. We build one branch labeled by the only element of $\{b\}$.

n_8 The node n_5 is labeled by "$\surd$" and is such that $H(n_5) \subset H(n_8)$, so we close the node, according to rule 3a.

After tree building, kernels correspond to the sets $H(n)$ for all nodes n labeled by "$\surd$". So in our example, the set of kernels is $\{\{a,b\},\{b,c\}\}$.

2.4 Algorithm adjustments

We now present the heuristics with which we adapt the initial algorithm.

2.4.1 Determination of an inconsistent subset

The implementation of our revision strategy requires the determination of the kernels of the collection $\mathcal{I}(I \cup E)$ of inconsistent subsets of $I \cup E$. The accesses to a collection $\mathcal{F}$ in section 2.3 are in our context the accesses to the set $\mathcal{I}(\mathcal{I} \cup E)$, that is the generation of an inconsistent subset of $I \cup E$. This generation can be achieved by the use of the Davis-Putnam procedure [20], providing that $I \cup E$ itself is inconsistent.

[2]This is the branching rule that transforms the tree into a directed acyclic graph.

[3]A node having more than one ancestor is a node upon which rule (3a) has been applied.

An inconsistent subset of $I \cup E$ consists of the set of unsatisfied clauses when inconsistency is detected (all literals in these clauses are assigned the **false** value). A leaf of the evaluation tree is reached when the empty clause is produced. We label each such leaves by the set F_i of unsatisfied clauses at this step. An inconsistent subset of a finite set of clauses is built by picking up an element in each set F_i. There is obviously no models for this set of clauses. Sets produced by this method can be very large. The larger they are, the more branches they will produce in the kernel construction tree. We propose heuristics, stemming from the syntactical structure of the problem, which allow a faster production of inconsistent subsets of the clause base. Moreover, these heuristics allow to minimize the inconsistent subsets sizes.

Proposition 2.1. *Let $E' \subseteq E$ such that $I \cup E'$ is inconsistent and such that $\forall E'' \subset E'$, $I \cup E''$ is consistent. Let $c_a \in E'$ and $c_a = \{a\}$. Then $\exists c \in I$, $c = \neg a \vee \neg l_i$, with l_i being any literal.*

Proposition 2.2. *$I_d \cup E$ is consistent.*

Consequence 2.3. *Let $E' \subseteq E$ such that $I \cup E'$ is inconsistent and $\forall E'' \subset E'$, $I \cup E''$ is inconsistent. Let $c_a \in E'$ and $c_a = \{a\}$. Then$\exists c \in I_c$, $c = \neg a \vee \neg l_i$, with l_i being any literal.*

Consequence 2.3 can be intuitively explained by the fact that an inconsistency corresponds to the exhausting of a domain. This exhausting is due to the combined action of constraints in I and initial prefered values in E. Based on propositions 2.1, 2.2 and on their consequence 2.3, our literal choice strategy for Davis-Putnam algorithm is as follows:

- choose literal in a monoliteral clause in I in priority (this is the classical monoliteral propagation heuristic),
- if there is not any monoliteral clause available in I, choose a monoliteral clause in E, according to propositions 2.1 and consequence 2.3 in order to produce monoliteral clauses in I.

For the choice of a monoliteral clause in E, we will choose the one which has the greatest number of occurences in I_c clauses, according to proposition 2.2. This is justified by the intuition developped in [18], that these clauses which are more often falsified have a greater probability to belong to minimal inconsistent subsets. Likewise, literals appearing more often in falsified clauses have a greater chance to belong to minimal-inconsistent subsets. Moreover, in case domains have the same extent, we are limiting the counting of literals from E to I_c because these literals have the same number of occurences in I_d. This allows a quicker count.

Expected gains from this strategy are double : First, choosing a literal from E in those appearing more frequently in binary negative clauses of I_c limits Davis-Putnam evaluation tree depth, generating less leaves and thus reducing the size of generated inconsistent subsets of $I \cup E$, while reducing the production time of these subsets. As a consequence, choosing a literal in E maximises the number of monoliteral clauses, thus allowing more often the use of monoliteral propagation, which efficiency has not to be proven anymore.

2.4.2 Optimizing kernel computing

Each kernel contains elements from either E or I, since for each kernel $N \in \mathcal{N}(\mathcal{I}(I \cup E))$, $\forall M \in \mathcal{I}(I \cup E)$, $M \cap N \neq \emptyset$, $M \cap I \neq \emptyset$ and $M \cap E \neq \emptyset$. We recall that in the context of revising E by I we wish to retain information in I and give up information in E responsible of the inconsistence. The first optimization consits in retaining only kernels containing elements from E only. This is possible due to following proposition.

Proposition 2.3. *$\exists N \in \mathcal{N}(\mathcal{I}(I \cup E))$ such that $N \cap I = \emptyset$.*

This proposition allows a pruning of the kernel tree. We already saw that each node is labeled by an inconsistent subset of $I \cup E$. According to 2.3, each node holds as many children as elements in its labeling inconsistent subset. As we're looking for kernels such that $N \cap I = \emptyset$, and we know by proposition 2.3 that they exist, we can simply ignore branches labeled by clauses from I.

The collection of kernels $N \in \mathcal{N}(\mathcal{I}(I \cup E))$ such that $N \cap I = \emptyset$ is denoted by $\mathcal{N}_E(\mathcal{I}(I \cup E))$. With this pruning technique, the new algorithm can be stated as follows:

Computation of $\mathcal{N}_E(\mathcal{I}(I \cup E))$. *Let $\mathcal{I}(\mathcal{I} \cup E)$ the collection of inconsistent subsets of $(I \cup E)$. The building tree T for kernels $\mathcal{N}(\mathcal{I}(I \cup E))$ is the smallest tree verifying the following properties :*

1. *Its root is labeled by "$\surd$" if $\mathcal{I}(I \cup E)$ is empty. Otherwise, its root is labeled by an element of $\mathcal{I}(I \cup E)$.*
2. *If n is a node in T, we define $H(n)$ as the set of branches labels on the path going from T root to n. If n is labeled by "$\surd$", it doesn't have any successor node in T. If is labeled by a set $\Sigma \in \mathcal{I}(I \cup E)$, then for each $\sigma \in \Sigma$ such as $\sigma \in E$ (according to proposition 2.3), n have a successor node n_σ*

linked to n by a branch labeled by σ. n_σ is labeled by a set $S \in \mathcal{I}(I \cup E)$ such that $S \cap H(n_\sigma) = \emptyset$ if such an S exists. If there is no such S, n_σ is labeled by "$\surd$".

Consequence 2.4. *Maximum depth of the computing tree of kernels $\mathcal{N}_E(\mathcal{I}(I \cup E))$ is $|E|$*[4].

3 An application in the framework of GIS

3.1 Importance for geographic applications

The general underlying objective of this work, is the improvement of analysis and management of the geographic information (GI), which makes an extensive use of "imperfect" data. Hence the *quality issue of GI* was our concern, and more specifically, the development of AI tools for imperfect GI. Through the word "imperfect", we address data which are imprecise (e.g. a height value may vary within an interval), uncertain (e.g. the vegetation type may be "traditional vine" or "mechanized") incomplete (e.g. some land parcels are informed, some other not).

When mixing several imperfect data in the purpose of analyzing some geographical phenomenon, the way these imperfections behave can rapidly become a complete mess. For example, a fuzzy approach for modeling the GI quality and its propagation suffers from the difficulty to fit well every kind of data. The resulting combination of each fuzzy model against a unique numerical scale leads to impose a total order on data items which are sometimes basically uncomparable. Hence the decision making process is not reliable.

Though, many geographical, physical,... knowledge can generally be used to constrain the imperfection. This kind of knowledge is often symbolical, such as default rules (in the Reiter's sense of: let's apply the rule as long as being not in contradiction with something else), but cannot be easily put into GIS.

These considerations lead us to privilege the symbolical representation of the GI data quality. For example, it is very easy to translate the following knowledge: if the water flows from A to B, the water height in A must be greater than in B. This paper illustrates the translation of several such knowledge pieces into propositional calculus, and the attempt to manage the imperfection by the means of revision. The counterpart is that the full set of formulae is inconsistent, and that any general approach to such a problem is NP-complete, and untractable even in simple and nice cases (see 5). But what is particular with the GI case is the 'G', that is there is a "space" beneath, all the objects have a location, and they are more or less distant to each other, hence influence each other differently and generally not at all, unless when near. As a consequence, we can assert a kind of meta assumption of "localness" about the propagation rules, but also about the propagation of "revisable" features: *if something goes wrong somewhere, the cause must be somewhere around.*

This conclusion was the startpoint of this work as a part of a European Community funded project (ESPRIT 27781), called REVIGIS (Knowledge Revision in GIS), whose purpose is to experiment, demonstrate and promote the use of non-classical logics within the GIS industry [12].

3.2 Knowledge representation of the flood study

The valley is partitioned into a set of compartments, denoted by $\mathcal{C}$. These are land plots delimited by natural obstacles like dykes, dry stone walls,... where the water height is constant within an interval $c =< c^-, c^+ >$ where the lower and upper bound, c^- and c^+ respectively, represent the minimal and the maximal submersion heigths of the compartment c.

3.2.1 The sources of information

From aerial pictures, we identify two kinds of hydraulic relations between some adjacent compartments. The flow relation denoted by f reflects the presence of an hydraulic link between two compartments with visible water flow. The hydrodynamic balance relation, denoted by e, reflects the presence of an hydraulic link between two compartments, but with no visible flow.

Each relation belonging to $\{e, f\}$ sets position constraints between the linked intervals bounds. This can be viewed as a relative position constraint between two intervals and can be expressed by the Allen's relations [2]. Since each Allen's relation can be translated into a set of four constraints on the intervals bounds [2, 10], we get a set of equalities and inequalities constraining the intervals bounds. This set is our first source of information.

The second source of information comes from land agricultural use. From this information, we establish initial estimations on the minimal and/or maximal submersion heights. We represent these estimations by a set of equalities E pointing out the values of the initial estimations.

[4] $|E|$ denotes the cardinal of set E.

3.2.2 Propositional encoding

The propositional encoding stems from the Bahia group's works [8]. This representation classically split into two aspects, the domain encoding and the constraint encoding. For now on, we consider the set $\mathcal{V} = \{X_1, \ldots, X_n\}$ of the n variables of the problem, that is, for each compartment, the set of lower and upper bounds of submersion. Each variable X_i is defined on a domain $D_i = \{a_{i1}, \ldots, a_{d_i}\}$ [5].

domain encoding For each variable X_i, we introdue d_i propositional variables A_{ij}, $1 \leq j \leq d_i$. A domain is then represented by the enumeration clause $A_{i1} \vee \ldots \vee A_{id_i}$, meaning that X_i takes its value in D_i, and by the excluding clauses which are negative binary clauses $\neg A_{ij} \vee \neg A_{ij'}$, $1 \leq j < j' \leq d_i$, specifying that X_i only takes one value. For each domain, we introduce one enumeration clause and $d_i(d_i - 1)/2$ mutual excluding clauses for the values of the domain. We denote by I_d this set of clauses representing the description of the domains.

constraint encoding We distinguish two kinds of constraints. First, the constraints from the set I, consisting in equalities and inequalities on two variables, the interval bounds. Each constraint belongmg to I implicitely defines a set of couples of permitted values for the two variables of the constraint. Since the variables domains are discrete, each constraint also defines a set of forbidden values. In the case where the variables domains intersect and when this intersection is large with respect to the domain size, coding the forbidden couples is more convenient because they generate a smaller number of formulas. Each forbidden couple is represented by a negative binary clause of the form $\neg A_{il} \vee \neg A_{jm}$ and we denote by I_c the set of such clauses.

We then consider the constraints from the set E, consisting in equalities specifying a preferentiel value for a variable. Each constraint of E is represented by a positive monoliteral clause and we denote by E the set of such clauses.

After the encoding step, the sets I_d, I_c and E have the properties specified in section 2.1, the revision strategy is performed and the following section presents the obtained experimental results.

4 Examples

We present an example involving three compartments. Tackled points cover the presentation of constraints in the example, their translation into propositionnal calculus and the unrolling of the $\mathcal{N}(\mathcal{I}(I \cup E))$ kernels calculus algorithm (KCA). The example is somewhat simplified comparing to the real data set for the sake of simplicity.

[5] in our problem the domains have same size.

4.1 Constraints

We consider three compartments[6] C_{61}, C_{64} and C_{286}. As showed in figure 2, we observe a flow going from C_{61} to C_{64}, and another flow going from C_{61} to C_{286}. We do not have any particular constraints between C_{64} and C_{286}. We recall that for a given compartment C_n we denote by C_n^+ (resp. C_n^-) the maximum (resp. minimum) submersion height. The translation of these flow relations in term of constraints on the interval bounds is as follows:

$$I = \left\{ \begin{array}{c} C_{61}^- \geq C_{286}^-,\ C_{61}^+ > C_{286}^-,\ C_{61}^+ \geq C_{286}^+, \\ C_{64}^- \leq C_{61}^-,\ C_{64}^- < C_{61}^+,\ C_{64}^+ \leq C_{61}^+, \\ C_{61}^- < C_{61}^+,\ C_{64}^- < C_{64}^+,\ C_{286}^- < C_{286}^+ \end{array} \right\}$$

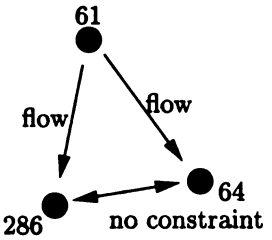


Figure 2: An example involving three compartments.

Moreover, we have the following initial assessments in centimeters (cm):

$$E = \left\{ \begin{array}{ll} C_{64}^- = 936, & C_{64}^+ = 966, \\ C_{286}^- = 909, & C_{286}^+ = 949, \\ C_{61}^- = 910, & C_{61}^+ = 930 \end{array} \right\}$$

4.2 Propositionnal calculus translation

We choose to discretize intervals every 50 cm[7], and we obtain the following translation (the full translation contains 122 clauses, so we only present the translation of one domain and one constraint). We recall that $A_{n,m}^-$ (resp. $A_{n,m}^+$) denotes the value m for the minimum (resp. maximum) submersion height of compartment C_n). The domain for the bound C_{286}^- is translated as follows:

[6] these compartments are extracted from the full data set of the application.

[7] In order to obtain a reproductible example in the context of this article, we choose this discretization value. In reality, we use a discretization value of 20cm.

$$A^-_{286,850} \vee A^-_{286,900} \vee A^-_{286,950} \vee A^-_{286,1000},$$
$$\neg A^-_{286,850} \vee \neg A^-_{286,900},$$
$$\neg A^-_{286,850} \vee \neg A^-_{286,950},$$
$$\neg A^-_{286,850} \vee \neg A^-_{286,1000},$$
$$\neg A^-_{286,900} \vee \neg A^-_{286,950},$$
$$\neg A^-_{286,900} \vee \neg A^-_{286,1000},$$
$$\neg A^-_{286,950} \vee \neg A^-_{286,1000}.$$

Other bounds have an indentical translation, with $C^+_{286} \in \{850, 900, 950, 1000\}$, $C^-_{61} \in \{850, 900, 950, 1000\}$, $C^+_{61} \in \{850, 900, 950, 1000\}$, $C^-_{64} \in \{900, 950, 1000, 1050\}$, $C^+_{64} \in \{900, 950, 1000\}$. The constraint $C^-_{61} \geq C^-_{286}$ is translated as

$$\neg A^-_{286,900} \vee \neg A^-_{61,850},$$
$$\neg A^-_{286,950} \vee \neg A^-_{61,850},$$
$$\neg A^-_{286,950} \vee \neg A^-_{61,900},$$
$$\neg A^-_{286,1000} \vee \neg A^-_{61,850},$$
$$\neg A^-_{286,1000} \vee \neg A^-_{61,900},$$
$$\neg A^-_{286,1000} \vee \neg A^-_{61,950},$$

Other constraints are similarly translated. All these clauses constitute the set I. Concerning the set E, initial assessments generate the following monoliteral clauses: $A^-_{286,900}$, $A^+_{286,950}$, $A^-_{61,900}$, $A^+_{61,950}$, $A^-_{64,950}$ and $A^+_{64,950}$.

4.3 Building the computation tree for $\mathcal{N}_E(\mathcal{I}(I \cup E))$

We present the kernel computation tree in figure 3. This tree is built using heuristics and pruning techniques previously exposed. We now explain this building (node numbers are the circled nodes in figure 3, and correspond to the breadth-first order of tree building).

The root node (1) is computed by calling the Davis-Putnam (DP) procedure as exposed in section 2.4.1. As exposed in section 2.4.2, we only keep tree branches which are labelled with elements of set E. So we have two branches under the root. We then build node (2). There is no reusable node label using the properties exposed in section 2.3, so we compute a new one. We then build node (3). We cannot reuse any node, so we try to compute a new one, but the DP procedure reply that the new set is consistent (that is $(I \cup E) \setminus \{A^-_{64,950}\}$ is consistent). So node (3) is labelled with "$\surd$", producing a first kernel. We now examine node (4), calling the DP procedure on the set $(I \cup E) \setminus \{A^+_{61,950}, A^-_{61,900}\}$. This produces a new inconsistent set, which isused to label node (4). Node (5) is closed because there is a shortest path from root to an end-node ("$\surd$") containing a branch labelled with $A^-_{64,950}$. Node (6) is closed for the same reasons as node (5). For node (7), the DP procedure tells us that $(I \cup E) \setminus \{A^+_{64,950}, A^-_{61,900}, A^+_{61,950}\}$ is consistent, giving us a second kernel.

We finally obtain two kernels: $\mathcal{N}_E(\mathcal{I}(I \cup E)) = \{\{A^-_{61,900}, A^+_{61,950}, A^+_{64,950}\}; \{A^-_{64,950}\}\}$.

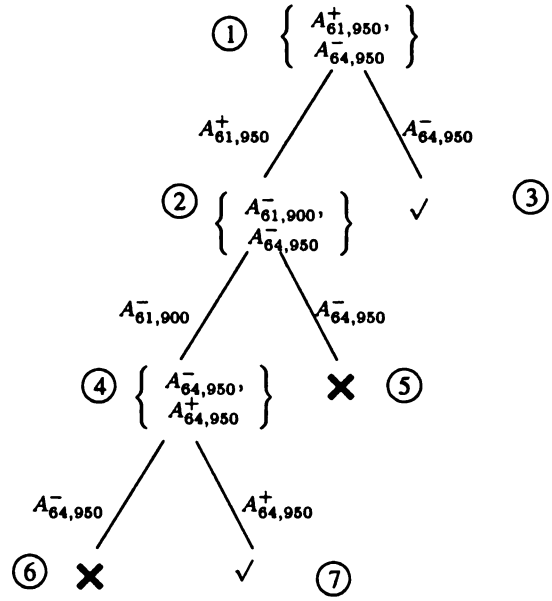


Figure 3: Example involving three compartments : kernels computing.

5 Experimental results

The kernel computation algorithm (KCA) has been implemented using C langage and the egcs 1.0.2 compiler with -O2 optimizations activated. Tests have been conducted using a PC equipped with a Pentium 166Mhz processor and 32Mo RAM. We worked on a large portion of the studied valley containing sixteen compartments. We tried to treat this portion as a whole and also we have partitionned the portion into overlapping packets of three or four adjacents compartments.This leds us to a set of fourteen packets. We compared our kernel computation algorithm to the following approaches:

ROBDD problem is represented using *Reduced Ordered Binary Decision Diagrams* (ROBDD) [6]. The program computes removed sets using the ROBDD, using similar techniques as in [4, 5, 16].

CONFLEX this is a flexible constraint satisfaction problem (flexible CSP) solver. Without going into out of topic details, we used two different distributions of preferences over discrete variables values. These two different cases are reffered as PCST and BCST. The CSP resolution algorithm is forward checking. We used a "flexible CSP" prototype programme [11, 14].

ATMS We used the notion of *nogoods* used in ATMS [15] in order to compute removed sets. The choosen algorithm is the one presented in [7]. It as to be said that the implementation of this algorithm is very space-consuming. Perhaps this point can be optimized.

Table 1 presents the size of the problem in its logical representation[8]. Table 2 presents for each algorithm

	full region	partition
number of variables	287	45
number of clauses	4097	457

Table 1: Size of the representation in propositional calculus for the test region.

the total computation time for the fourteen packets and also the mean time for one packet. Concerning KCA, we recall that this algorithm — unlike the other ones — doesn't give a solution but identifies initial preferred assessments (from E) to retract in order to restore consistency . At this point of the test, KCA is

	total time (s)	mean time (s)
Conflex PCST	9"52	0"680
Conflex BCST	9"40	0"670
ROBDD	5"29	0"370
ATMS	stopped (1h)	
KCA	0"19	0"013

Table 2: Performance results on a partition of a portion of the valley

clearly the quickest, ROBDDs are fast enough, but very memory consuming, and cannot be applied to more than a 3–4 compartments. CONFLEX seems to have acceptable run times, but the control of the flexibility is not much "flexible": it is always some kind of a-priori declining function. Table 3 presents results for each algorithm the results on the whole choosed portion of the valley. This test has been conducted on a more powerfull machine (Pentium II with 256Mo RAM). Neither algorithm reached the end of its computation, but it has to be noted that KCA gave some results before failing. It found four kernels with a cardinality of 3, since KCA uses a breadth-first tree construction.

	stop condition	comments
Conflex	ST (15 days)	NR
ROBDD	ME	NR
ATMS	ST (15 days)	NR
KCA	ME	4 kernels

ME: memory exhausting, ST: stopped, NR: no results

Table 3: Results of the full test.

6 Conclusion

In this paper, we present a logical representation of a geographic problem involving two conflicting sources. This leads to deal with a revision problem with a large amount of data, which requires a suitable adjustment of existing revision methods: we propose one based on inconsistency minimization. After a characterization of removed sets by means of kernels, we adapt the Reiter's algorithm for diagnosis, in order to compute the kernels. The experimental results are better than those previously obtained with other methods.

This approach could be successfully applied to other applications in geophysics or in demography... with the same constraint structure than in the present application: a CSP graph, given by the underlying space, and domain constraints consisting in intervals. In a future work, we intend to use the specific nature of this geographic problem, in two ways:

- to order the kernels according to different orders, eg.: the distance between compartments;
- to take advantage of the spatial structure for dividing the problem into simpler ones, hence to reduce the complexity.

Acknowledgements

This work was supported by European Community project ESPRIT 27781 REVIGIS.

References

[1] Alchourrón, Gärdenfors, and Makinson. On the logic of theory change : Partial meet contrac-

[8] Not counting the variables added for ROBDD and ATMS.

tion and revision functions. *J. of Symbolic Logic*, 50(2):510–530, 1985.

[2] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[3] B. Bessant, E. Grégoire, P. Marquis, and L. Sais. Combining nonmonotonic reasoning and belief revision : a practical approach. In F. Giunchiglia, editor, *International Conference on Artificial Intelligence : Methodology, Systems, Applications*, volume 1480 of *Lecture Notes in Artificial Intelligence*, pages 115–128. Springer-Verlag, 1998.

[4] Fabrice Bouquet and Philippe Jégou. Solving over-constrained CSP using weighted OBDDs. In Michael Jampel, Eugene Freuder, and Michael Maher, editors, *Over-Constrained Systems*, volume 1106 of *Lecture Notes in Computer Science*, pages 293–308. Springer-Verlag, 1996.

[5] Fabrice Bouquet and Philippe Jégou. Ordres et diagrammes de décision binaire ordonné. Rapport technique 275, Laboratoire d'Informatique de Marseille, Marseille, 1998.

[6] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on computers*, C-35(8):677–691, Aout 1986.

[7] Thierry Castell, Claudette Cayrol, Michel Cayrol, and Daniel Le Berre. Using the Davis and Putnam procedure for an efficient computation of prefered models. In W. Wahlster, editor, *ECAI96*. John Wiley and Sons, Ltd, 1996.

[8] Jean-Jacques Chabrier, Jacqueline Chabrier, Olivier Palmade, Michel Cayrol, Antoine Rauzy, Philippe Ezequel, Jin Kao Hao, Gérard Plateau, Hachemi Bennaceur, Christien Bessière, Philippe Janssen, Marie-Catherine Vilarem, Belaid Benhamou, Philippe Jegou, Laurent Oxusoff, Lakdar Sais, and Pierre Siegel. Étude comparative des trois formalismes en calcul propositionnel, Projet Bahia. In *Actes des 4èmes journées nationales PRC-GDR IA*, pages 239–314, Marseille, Octobre 1992. Teknéa.

[9] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactual. *Artificial Intelligence*, 57:227–270, 1992.

[10] Jérôme Euzenat. An algebraic approach to granularity in qualitative time and space representation. In *Proc. 14th IJCAI*, pages 894–900, Montreal (CA), 1995.

[11] Hélène Fargier, Didier Dubois, and Henri Prade. Problèmes de satisfaction de contraintes flexibles : une approche égalitariste. *Revue d'Intelligence Artificielle*, 9:311–354, 1995.

[12] Peter Fisher, Andrew U. Frank, Ben Gorte, Robert Jeansoulin, Jenny Lingham, Martin Molenaar, Navratil, Anthony J. Roy, Odile Papini, John Stell, Sabine Timpf, Maurits van der Vlugt, Michael F. Worboys, and Eric Wurbel. Revigis project : First step final report, http://www.cmi.univ-mrs.fr/REVIGIS. Technical report, Esprit project 27781, May 1999.

[13] Laurent Garcia. *Raisonnement local dans les bases de connaissances ordonnées : Applications au traitement des incohérences et au raisonnement plausible.* Thèse de doctorat, Université Paul Sabatier de Toulouse, Novembre 1998.

[14] INRA, Laboratoire de Biométrie et d'IA, BP 27, 31326 Castanet Tolosan cedex, France. *CON'FLEX 1.2, Manuel de l'utilisateur*, Janvier 1998.

[15] J. De Kleer. An assumption-based tms. *Artificial Intelligence*, 28:127–162, 1986.

[16] Marie-Christine Lagasquie-Schiex. *Contribution à l'étude des relations d'inférence non-monotone combinant inférence classique et préférences.* PhD thesis, Université Paul Sabatier, Toulouse, IRIT, Université Paul Sabatier, 118 route de Narbonne, 31062 Toulouse Cedex, Décembre 1995.

[17] Paolo Liberatore and Marco Schaerf. The complexity of model checking for belief revision and update. In *AAAI'96*, pages 556–561, 1996.

[18] Bertrand Mazure. *De la satisfaisabilité à la compilation de bases de connaissances propositionnelles.* Thèse de doctorat, Université d'Artois, Janvier 1999.

[19] Bernhard Nebel. Syntax based approaches to belief revision. In Peter Gärdenfors, editor, *Belief Revision*, pages 52–88. Cambridge University Press, UK, 1992.

[20] Odile Papini. A complete revision function in propositionnal calculus. In B. Neumann, editor, *Proceedings of ECAI92*, pages 339–343. John Wiley and Sons. Ltd, 1992.

[21] Damien Raclot and Christian Puech. Photographies aériennes et inondation : globalisation d'informations floues par un système de contraintes pour définir les niveaux d'eau en zone

inondée. *Revue internationale de géomatique*, 8(1):191–206, Février 1998.

[22] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[23] Ralph W. Wilkerson, Russel Greiner, and Barbara A. Smith. A correction to the algorithm in reiter's theory of diagnosis. *Artificial Intelligence*, 41:79–88, 1989.

[24] M. A. Williams and D. Williams. A belief revision system for the world wide web. In *Proceedings of the IJCAI workshop of the Future of Artificial Intelligence and the Internet*, pages 39–51, 1997.

Representation Formalisms

APPROXIMATE OBJECTS AND APPROXIMATE THEORIES

John McCarthy
Computer Science Department
Stanford University
Stanford, CA 94305
jmc@cs.stanford.edu
http://www-formal.stanford.edu/jmc/

Abstract

We propose to extend the ontology of logical AI to include approximate objects, approximate predicates and approximate theories. Besides the ontology we treat the relations among different approximate theories of the same phenomena.

Approximate predicates can't have complete if-and-only-if definitions and usually don't even have definite extensions. Some approximate concepts can be refined by learning more and some by defining more and some by both, but it isn't possible in general to make them well-defined. Approximate concepts are essential for representing common sense knowledge and doing common sense reasoning. Assertions involving approximate concepts can be represented in mathematical logic.

A sentence involving an approximate concept may have a definite truth value even if the concept is ill-defined. It is definite that Mount Everest was climbed in 1953 even though exactly what rock and ice is included in that mountain is ill-defined. Likewise, it harms a mosquito to be swatted, although we haven't a sharp notion of what it means to harm a mosquito. Whatif(x,p), which denotes what x would be like if p were true, is an important kind of approximate object.

The article treats successively approximate objects, approximate theories, and formalisms for describing how one object or theory approximates another.

Our discussion will be adequate if it has as much clearness as the subject matter admits of, for precision is not to be sought for alike in all discussions, any more than in all the products of the crafts.
—Aristotle, *Nicomachean Ethics*

1 Introduction

We propose to extend the ontology of logical AI to include approximate objects, approximate predicates and approximate theories. Besides the ontology we discuss relations among different approximations to the same or similar phenomena.

The article will be as precise as we can make it. We apply Aristotle's remark to the approximate theories themselves. The article treats three topics.

Approximate objects These are not fully defined, e.g. the wishes of the United States. They may be approximations to more fully defined objects or they may be intrinsically approximate (partial). We give lots of examples, because we don't have precise definitions.

Approximate theories These often involve necessary conditions and sufficient conditions but lack conditions that are both necessary and sufficient.

Relations among approximate entities It is often necessary to relate approximate entities, e.g. objects or theories, to less approximate entities, e.g. to relate a theory in which one block is just on another to a theory in which a block may be in various positions on another.

In principle, AI theories, e.g. the original proposals for situation calculus, have allowed for *rich* entities which could not be fully defined. However, almost

all theories used in existing AI research has not taken advantage of this generality. Logical AI theories have resembled formal scientific theories in treating well-defined objects in well-defined domains. Human-level AI will require reasoning about approximate entities.

Approximate predicates can't have complete *if-and-only-if* definitions and usually don't even have definite extensions. Some approximate concepts can be refined by learning more and some by defining more and some by both, but it isn't possible in general to make them well-defined. Approximate concepts are essential for representing common sense knowledge and doing common sense reasoning. In this article, assertions involving approximate concepts are represented in mathematical logic.

A sentence involving an approximate concept may have a definite truth value even if the concept is ill-defined. It is definite that Mount Everest was climbed in 1953 even though exactly what rock and ice is included in that mountain is ill-defined. We discuss the extent to which we can build solid intellectual structures on such swampy conceptual foundations.

Quantitative approximation is one kind considered—but not the most interesting or the kind that requires logical innovation. Fuzzy logic involves a semi-quantitative approximation, although there are extensions as mentioned in [Zad99].

For AI purposes, the key problem is relating different approximate theories of the same domain. For this we use mathematical logic fortified with contexts as objects. Further innovations in logic may be required to treat approximate concepts as flexibly in logic as people do in thought and language.

Looked at in sufficient detail, all concepts are approximate, but some are precise enough for a given purpose. McCarthy's weight measured by a scales is precise enough for medical advice, and can be regarded as exact in a theory of medical advice. On the other hand, McCarthy's purposes are approximate enough so that almost any discussion of them is likely to bump against its imprecision and ambiguity.

Many concepts used in common sense reasoning are imprecise. Here are some questions and issues that arise.

1. What rocks and ice constitute Mount Everest?

2. When can it be said that one block is on another block, so that
   On(Block1, Block2) may be asserted?

   Let there be an axiomatic theory in situation calculus in which it can be shown that a sequence of actions will have a certain result. Now suppose that a physical robot is to observe that one block is or is not on another and determine the actions to achieve a goal using situation calculus. It is important that the meanings of On(Block1, Block2) used in solving the problem theoretically and that used by the robot correspond well enough so that carrying out a plan physically has the desired effect. How well must they correspond?

3. What are the logical relations between different logical specifications of an approximate object?

4. What are the relations between different approximate logical theories of a domain?

We claim

1. The *common sense informatic situation* often involves concepts which cannot be made precise. This is a question of the information available and not about computation power. It is not a specifically human limitation and will apply to computers of any possible power. This is not a claim about physics; it may be that a discoverable set of physical laws will account for all phenomena. It is rather a question of the information actually available about particular situations by people or robots with limited opportunities to observe and compute.

2. Much pedantry in science and in philosophy results from demanding *if-and-only-if* definitions when this is inappropriate.

3. None of the above objects and predicates admits a completely precise **if-and-only-if** definition.

4. Formalized scientific theories, e.g. celestial mechanics and the blocks world often have precise concepts. They are necessary tools for understanding and computation. However, they are imbedded in common sense knowledge about how to apply them in world of imprecise concepts. Moreover, the concepts are precise only within the theory. Most AI theories have also had this character.

5. Almost all concepts are approximate in the full common sense informatic situation. However, many are precise, i.e. have if-and-only-if definitions in particular contexts.

6. The human reasoning processes involving common sense knowledge correspond only partly to

the mathematical and logical reasoning processes that humans have used so successfully within scientific and mathematical theories.

7. **Nevertheless, mathematical logic is an appropriate tool for representing this knowledge and carrying out this reasoning in intelligent machines.** The use of logic will certainly require adaptations, and the logic itself *may require* modifications.

8. Key tools for common sense reasoning by machines will be *approximate concepts* and *approximate theories*. These are in addition to *formalized non-monotonic reasoning* and *formal theories of context*.

9. The most important notion of approximation for common sense reasoning is not the familiar numerical approximation but a new kind of logical approximation appropriate for common sense reasoning. These mostly differ from the approximations of fuzzy logic.

In the subsequent sections of this article, tools will be proposed for reasoning with approximate concepts.

The article treats successively *approximate objects*, *approximate theories*, and formalisms for describing how one object or theory approximates another.

2 What kinds of approximate concepts are there?

Concepts can be approximate in at least two ways.

On one hand, a concept may approximate another more definite but incompletely known concept. This situation is prevalent with *natural kinds*. Lemons are a definite species, but no-one knows all about them. In particular, a child may be barely able to tell lemons from other yellow fruit but nevertheless has a concept of lemon that may be adequate to find the lemons in the supermarket. The child can improve its concept of lemon by learning more. The fact that there isn't a continuum of fruits ranging from lemons to grapefruit is an important part of the fact that lemons form a natural kind. This fact also makes it possible for biologists to learn specific facts about lemons, e.g. to sequence lemon DNA.

On the other hand, the legal concept of a person's taxable income is refined by defining more. *Taxable income* is partly a natural kind. A person's concept of his own taxable income is an approximation to the less approximate legal concept. He could learn more or it could be defined more as the legal concept is defined more. However, learning more about the legal concept eventually reaches a point where there is no further refinement on which people thinking independently will agree. There isn't a true notion of taxable income for economists to discover.

My concept of my taxable income and even my tax accountant's concept of my taxable income has both aspects. More can be learned about what deductions are allowed, and also the concept gets refined by the courts.

Here are some examples.

2.1 What if?

Whatif(p, x) is what x would be if p were true. Examples: (1) John McCarthy if he had gone to Harvard rather than to Caltech as an undergraduate. (2) My car if it hadn't been backed into today. (3) The cake I would have baked if I had known you were coming. (4) What the world would be like today if Pickett's charge had been successful. (5) What would have happened if another car had come over the hill when you passed that Mercedes just now.

Whatif(p, x) is an intrinsically approximate concept. How approximate depends on p and x. "What if another car had come over the hill when you passed" is much less approximate than"What if wishes were horses". [CM99] treats useful counterfactual conditional sentences and gives many examples.

Whatif can serve as the foundation for other concepts, including counterfactual conditional sentences and statements of causality. [CM99] treats useful counterfactual conditional sentences and gives many examples.

Fiction provides an interesting class of approximate objects and theories, especially historical fiction, in which the author tries to fit his characters and their lives into a background of historical fact. Common sense knowledge tells us that Sherlock Holmes would have had a mother, but Conan Doyle does not provide us with a name. A definite address is given, but there was never a house there corresponding to Doyle's description. The author need only define his world to a point that lets the reader answer the questions the author wants the reader to ask.

2.2 Mount Everest.

It is clear that we cannot hope to formulate a useful definition of Mount Everest that would tell about every rock whether it is part of the mountain. However, we

might suppose that there is a truth of the matter for every rock even though we cannot know it. Our axioms would then be weaker than the truth. The question would be settled for some rocks and not for others.

Not even this is appropriate. The concept of the territory of Mount Everest may be further refined in the future—and refined in incompatible ways by different people. If we suppose that there is a truth about what rocks are part of the mountain, then the people refining it in different ways would get to argue fruitlessly about which definition is getting closer to the truth. On the other hand, there is a truth of the matter, which may someday be discovered, about whether Mallory and Irvine reached the summit in 1924.

Consider two theories of mountain climbing, $T1$ and $T2$. Besides these theories, there is $T3$ based on plate tectonics that tells us that Everest is still getting higher.

In the simpler theory $T1$, there is a list of names of mountains paired with lists of climbing expeditions or names of climbers. As a logical theory it would have sentences like.

$$Climbed(Everest, 1953, \{\text{Hillary, Tenzing}\}). \quad (1)$$

The larger theory $T2$ contains routes up the mountain of the various parties. Routes are approximate entities.

$T1$ is an approximation to $T2$, but $T1$, may be regarded as not approximate at all. In particular, it can be complete, e.g. it decides any sentence in its limited language.

$T1$ and $T2$ may be related using formalized contexts as in [McC93a] or [MB97], but we won't do that here.

One approximate theory may be less approximate than another. We want to discuss relations between sentences in a theory and sentences in a less approximate theory. It makes the ideas neater if we imagine that there are true and complete theories of the world even if they're not finitely expressible and none of the language of any of them is known. This permits regarding approximating the world as a case of one theory approximating another. If this is too platonic for your taste, you can regard approximating the world as a different notion than that of one theory approximating another.

There are other approximate theories involving Mount Everest.

One such theory that lists names of mountains and the continents containing them. Thus we have In(Annapurna, Asia). A less approximate theory gives countries, e.g. In(Annapurna, Nepal). A still less approximate theory gives locations, e.g. Location(Annapurna) = $(28°32', 83°53')$.

2.3 The wants and actions of the United States

In 1990 the United States wanted Iraq to withdraw from Kuwait. Evidence for this proposition was provided by the statements of U.S. officials and sending troops to Saudi Arabia. It was correctly inferred from this proposition that the U.S. would do something to implement its desires. This inference was made with only an approximate notion of "the US wants".

Nevertheless, the facts can be expressed by formulas like the following.

$$\text{Says}(\text{President}(\text{USA}), x) \rightarrow \text{Says}(\text{USA}, x), \quad (2)$$

$$\text{Says}(\text{President}(\text{USA}), \text{Wants}(\text{USA}, \text{Leaves}(\text{Iraq}, \text{Kuwait}))), \quad (3)$$

$$\text{Says}(\text{USA}, \text{Wants}(\text{USA}, \text{Leaves}(\text{Iraq}, \text{Kuwait}))), \quad (4)$$

$$\text{Says}(entity, \text{Wants}(entity, x)) \rightarrow \text{wants}(entity, x), \quad (5)$$

$$\text{wants}(\text{USA}, \text{Leaves}(\text{Iraq}, \text{Kuwait})), \quad (6)$$

$$\text{wants}(x, y) \rightarrow (\exists z)(\text{Does}(x, z) \wedge \text{Achieves}(z, y)). \quad (7)$$

From these we infer

$$(\exists z)(\text{Does}(\text{USA}, z) \wedge \text{Achieves}(z, \text{Leaves}(\text{Iraq}, \text{Kuwait})). \quad (8)$$

We have not introduced all the necessary qualifications, and we have not used a proper theory of actions. There also should be some more theory of Wants, Says, and Does.

Someone with a sufficiently detailed knowledge of events in the Middle East and of the American decision making community might not need "The US wants ...", because he could work directly with the

various decision makers and the motivations and effects of their actions. The rest of us must make do with more approximate concepts. This will apply even more to future students of 20th century history.

A fuzzy logic theory would take "The US wants" for granted and concentrate on "The US moderately wants" and "The US strongly wants".

2.4 The Blocks World as an Approximate Theory

The usual AI situation calculus blocks world has a propositional fluent $\mathrm{On}(x, y)$ asserting that block x is on block y. We can assert $\mathrm{Holds}(\mathrm{On}(x, y), s)$ about some situation s and have the action $\mathrm{Move}(x, y)$ that moves block x on top of block y.

Suppose this formalism is being used by a robot acting in the real world. The concepts denoted by $\mathrm{On}(x, y)$, etc. are then approximate concepts, and the theory is an approximate theory. Our goal is to relate this approximate theory to the real world. Similar considerations would apply if we were relating it to a more comprehensive but still approximate theory.

We use formalized contexts as in [McC93b] and [MB97]. and let Cblocks be a blocks world context with a language allowing $On(x, y)$, etc.

$\mathrm{Holds}(\mathrm{On}(x, y), s)$ is approximate in at least the following respects.

- In the original intended interpretation of the situation calculus, a situation s is a snapshot of the world at a given time. According to the theory of relativity, distant simultaneity is ill-defined. This is the least of our worries.
- Whether block x is on block y may be ambiguous in the real world. Block x may be partly on and partly off. We can handle the relation by sentences

$$\begin{aligned}&\mathrm{Cond1}(s) \rightarrow \mathrm{Ist}(\mathrm{Cblocks}, \mathrm{Holds}(\mathrm{On}(x, y), s))\\ &\mathrm{Cond2}(s) \rightarrow \mathrm{Ist}(\mathrm{Cblocks}, \mathrm{Holds}(\mathrm{Not\ On}(x, y), s)).\end{aligned} \tag{9}$$

  $\mathrm{Cond1}(s)$ and $\mathrm{Cond2}(s)$ are respectively conditions in the outer context on the situation s that x shall be on y and x shall not be on y in the context Cblocks. These need not be the negations of each other, so it can happen that it isn't justified to say either that x is on y or that it isn't. $\mathrm{Cond1}(s)$ and $\mathrm{Cond2}(s)$ need not be mutually exclusive. In that case the theory associated with Cblocks would be inconsistent. However, unless there are strong *lifting rules* the inconsistency within Cblocks cannot infect the rest of the reasoning.

Notice that the theory in the context Cblocks approximates a theory in which blocks can be in different orientations on each other or in the air or on the table in quite different sense than numerical approximation.

2.5 Relating two blocks world theories

Our previous blocks world theory T1 uses $\mathrm{Holds}(\mathrm{On1}(b1, b2), s)$. Our less approximate new theory T2 uses $\mathrm{Holds}(\mathrm{On2}(b1, b2, d), \sigma)$ where d is a displacement of $b1$ from being centered on $b2$. Since T2 has another parameter for On, many situations σ can correspond to a single situation s in T1.

We may have the relations

$$\begin{aligned}&\mathrm{Holds}(\mathrm{On2}(b1, b2, d)), \sigma)\\ &\rightarrow \mathrm{Holds}(\mathrm{On1}(b1, b2)), St1(\sigma))\\ &\mathrm{Holds}(\mathrm{On1}(b1, b2)), \mathrm{St1}(\sigma))\\ &\rightarrow (\exists d)\mathrm{Holds}(\mathrm{On2}(b1, b2, d)), \sigma).\end{aligned} \tag{10}$$

Here $\mathrm{St1}(\sigma)$ is the T1-situation corresponding to σ. For simplicity we are assuming that every T2-situation has a corresponding T1-situation.

T2 is a tiny step from T1 in the direction of the real world.

Suppose a robot uses T1 as a theory of the blocks world and takes actions accordingly, but the real world corresponds to T2. This is quite a simplification, but maybe it has enough of the right formal properties.

The simplest case is where there are two blocks, and the initial situation is represented by

$$\{\mathrm{Holds}(\mathrm{On1}(B1, \mathrm{Table}), S0), \mathrm{Holds}(\mathrm{On1}(B2, \mathrm{Table}), S0)\} \tag{11}$$

in T1, but the real world facts are

$$\begin{aligned}&\{\mathrm{Holds}(\mathrm{On2}(B1, \mathrm{Table}, 3.0), 2.1\,cm),\\ &\mathrm{Holds}(\mathrm{On}(B2, \mathrm{Table}, 4.5), 3.2\,\mathrm{cm})\}\end{aligned} \tag{12}$$

where

$$S0 = \mathrm{St1}(\sigma 0). \tag{13}$$

The goal is $\mathrm{Holds}(\mathrm{On}(B1, B2)$, which might also be written as $(\lambda s)\mathrm{Holds}(\mathrm{On}(B1, B2), s)$. Anyway the robot infers that the appropriate action is $\mathrm{Move}(B1, B2)$ and infers that

$$\mathrm{Holds}(\mathrm{On1}(B1, B2), \mathrm{Result}(\mathrm{Move1}(B1, B2), S0)), \tag{14}$$

where we are omitting various qualifications.

In T2, the form of an action is Move2($b1, b2, d$), and the effect of a move action is given by

$$\text{Holds}(\text{On}(b1, b2, d), \text{Result2}(\text{Move2}(b1, b2, d), \sigma)). \tag{15}$$

The translation of a move action in T1 to a move action in T2 may be given by

$$\begin{aligned} &\text{Action12}(\text{Move1}(b1, b2), St1(\sigma)) \\ =\ &\text{Move2}(b1, b2, \text{Displacement}(b1, b2, \sigma)). \end{aligned} \tag{16}$$

The key point is that the move in T2 corresponding to a move in T1 depends on the blocks being moved and also on the situation.

The success of the one step plan worked out in T1 in the less approximate world T2 is expressed by

$$\begin{aligned} &St1(\text{Result2}(\text{Action12}(\text{Move1}(B1, B2), St1(\sigma 0)))) \\ =\ &\text{Result1}(\text{Move}(B1, B2), St1(\sigma 0)). \end{aligned} \tag{17}$$

The success of multi-step plans would be expressed by longer correspondence formulas.

These are commutativity relations.

2.6 Temporary entities; wealth and welfare

In natural language, the present tense of the verb "to be" is used for asserting an intrinsic property of an entity and for asserting a property that is expected to hold long enough to provide a constant background for other events under discussion.

The wealth or welfare of a human or animal is such a temporary property.

The welfare of a mosquito over a short time is definable. It harms the mosquito to be squashed and helps it if it finds exposed skin from which to extract blood. Over a year the welfare of an individual mosquito is not definable. If it is to be defined, the concepts, e.g. descendants, will be quite different.

This suggest using contexts. We have

$$c(\text{Today}) : \text{Harms}(\text{Swat}(M_{1073543907}), M_{1073543907}) \tag{18}$$

as a proposition about this particular mosquito.

A person's wealth at a given time can be measured as an amount of money. His wealth increases as he is paid and decreases as he spends money. However, over a period of 10,000 years, the wealth or welfare of this individual is undefined.

Nevertheless, wealth and welfare are useful concepts.

Fiction provides an interesting class of approximate objects and theories, especially historical fiction, in which the author tries to fit his characters and their lives into a background of historical fact. Common sense knowledge tells us that Sherlock Holmes would have had a mother, but Conan Doyle does not provide us with a name. A definite address is given, but there was never a house there corresponding to Doyle's description. The author need only define his world to a point that lets the reader answer the questions the author wants the reader to ask.

2.7 States of motion

These present a general reason for using approximate concepts.

Suppose a robot is walking from here to there in discrete steps.

$$\text{Holds}(\text{Walking}(R2D2, \text{Here}, \text{There}), s)$$

describes a situation that can persist. Sentences giving the robot's position at a given time must be frequently updated. An important human reason for forming an approximate concept is to get a fluent that will persist. This reason will also apply to robots but perhaps to a lesser extent, since computers can perform more frequent updating than can people.

2.8 Folk physics and folk psychology as approximate theories

For example, the concept of "X believes P" is approximate both in the criterion for belief and in what is the object of a belief. These notions will also be approximate for robots.

Much of the criticism of folk psychology may come from demanding that it be more precise than is reasonable. Aristotle's aphorism applies here.

3 Propositional approximate theories

Many topics take an especially simple form when one uses propositions instead of predicates—and accepts the reduced expressivity.

Here is one approach to defining approximate propositional theories.

Let reality, e.g. the situation in a room, be given by the values of the propositional variables $r_1, \ldots, r_n$. Assume that reality is not directly observable. n may be very large, e.g. like Avogadro's number.

Let the values of the propositions $o_1, \ldots, o_k$ be observable. They are functions of reality given by

$$o_i = \mathcal{O}_i(r_1, \ldots, r_n),$$

where k is a modest number corresponding to how many bits we can actually observe.

We suppose that we want to know the values of $q_1, \ldots, q_l$, which are related to reality by

$$q_i = \mathcal{Q}_i(r_1, \ldots, r_n),$$

where l is also a modest number.

An approximate theory $\mathcal{AT}$ is given by functions $\mathcal{Q}'_i(o_1, \ldots, o_k)$, i.e. $\mathcal{AT}$ undertakes to give what we want to know in terms of the observations.

If we are lucky in how reality turns out, the $\mathcal{Q}'$ functions correspond to the $\mathcal{Q}$ functions, i.e.

$$Lucky(r_1, \ldots, r_n) \rightarrow q_i = \mathcal{Q}'_i(o_1, \ldots, o_k)$$

for $i = 1, \ldots, l$, i.e.

$$Lucky(r_1, \ldots, r_n) \rightarrow [\mathcal{Q}_i(r_1, \ldots, r_n) \equiv \mathcal{Q}'_i(\mathcal{O}_1(r_1, \ldots, r_n), \ldots \mathcal{O}_k(r_1, \ldots, r_n))].$$

If we are very fortunate we may be able to know when we are lucky, and we have

$$KnowLucky(o_1, \ldots, o_k) \rightarrow Lucky(r_1, \ldots, r_n).$$

At the moment, we have no useful propositional approximate theories in mind, and the reader should remember Einstein's dictum "Everything should be made as simple as possible—but not simpler."

3.1 Approximate Theories of Digital Circuits

Consider first combinational circuits built from logic elements, i.e. without bridges and other sneak paths. The logical elements are treated as boolean functions, defined by their truth tables. The theory defines the behavior of any circuit in terms of composition of the functions associated with the logical elements. The outputs of a circuit are given by the theory for any combination of boolean inputs. Fan-in and fan-out restrictions are outside the logical theory, as are timing considerations.

Now consider sequential circuits including flipflops. Now what happens is defined only for some combinations of inputs. For example, the behavior of a D flipflop is not defined when its 0 and 1 inputs are given the same value, whether that value be 0 or 1. The behavior is only defined when the inputs are opposite.

The manufacturer does not say what will happen when these and other restrictions are not fulfilled, does not warrant that two of his flipflops will behave the same or that a flipflop will retain whatever behavior it has in these forbidden cases.

This makes the concept of D flipflop itself approximate, perhaps not in the same sense as some other approximate theories.

Thus the theory of sequential circuits is an approximate theory, and it is not an approximation to a definite less approximate theory of purely digital circuits. This is in spite of the fact that there is (or can be) an electronic theory of these digital circuits which describes their behavior. In that theory one D flipflop is different from another and changes its behavior as it ages.

4 When an approximate concept becomes precise

Suppose an approximate concept represented by a predicate $p(x)$ has a sufficient condition $suff(x)$ and a necessary condition $nec(x)$. Thus we have

$$(\forall x)(suff(x) \rightarrow p(x)), \text{and} \quad (\forall x)(p(x) \rightarrow nec(x)). \tag{19}$$

In general the sufficient and the necessary conditions will not coincide, i.e. we will not have

$$(\forall x)(\text{nec}(x) \equiv \text{suff}(x)). \tag{20}$$

However, they made coincide with some restriction on x, i.e. we may have

$$(\forall x)(\text{special}(x) \rightarrow (\text{nec}(x) \equiv \text{suff}(x)). \tag{21}$$

Another way an approximate concept may become definite is by a mapping from the space in which it is first formalized into more restricted space. We'll combine specialization with mapping in

$$(\forall x)(\text{special}(x) \rightarrow (\text{nec}(f(x)) \equiv \text{suff}(f(x)), \tag{22}$$

where the function f maps a subset of the original domain into a specialized domain in which the concept $p(x)$ becomes definite.

5 Conclusions, remarks, and acknowledgements

The present paper is exploratory. The large number of approximate concepts we discuss are approximate in different ways, but we don't have a classification.

Many of the applications of approximate objects will result from theories connecting different levels and kinds of approximation.

I thank Eyal Amir, Johan van Benthem, Saša Buvač, Tom Costello, Aarati Parmar for helpful comments.

This research has been partly supported by ARPA contract no. USC 621915, the ARPA/Rome Laboratory planning initiative under grant (ONR) N00014-94-1-0775 and ARPA/AFOSR under (AFOSR) grant # F49620-97-1-0207.

References

[CM99] Tom Costello and John McCarthy. Useful Counterfactuals[1]. *Electronic Transactions on Artificial Intelligence*, 1999. submitted 1999 July.

[MB97] John McCarthy and Saša Buvač. Formalizing context (expanded notes). In A. Aliseda, R.J. van Glabbeek, and D. Westerståhl, editors, *Computing Natural Language*. Center for the Study of Language and Information, Stanford University, 1997.

[McC93a] John McCarthy. Notes on Formalizing Context[2]. In *IJCAI-93*, 1993.

[McC93b] John McCarthy. Notes on formalizing context. In *IJCAI93*, 1993. Available as http://www-formal.stanford.edu/jmc/context.html.

[Zad99] Lotfi A. Zadeh. From computing with numbers to computing with words—from manipulation of measurements to manipulation of perceptions. *IEEE Transactions on Circuits and Systems—I. Fundamental theory and applications*, 45(1):105–119, January 1999.

[1] http://www-formal.stanford.edu/jmc/counterfactuals.html
[2] http://www-formal.stanford.edu/jmc/context.html

Iterated Belief Change in the Situation Calculus

Steven Shapiro
Dept. of Computer Science
University of Toronto
Toronto, ON
M5S 3G4 Canada
steven@ai.toronto.edu

Maurice Pagnucco
Dept. of Computing
Macquarie University
NSW 2109
Australia
morri@ics.mq.edu.au

Yves Lespérance
Dept. of Computer Science
York University
Toronto, ON
M3J 1P3 Canada
lesperan@cs.yorku.ca

Hector J. Levesque
Dept. of Computer Science
University of Toronto
Toronto, ON
M5S 3G4 Canada
hector@ai.toronto.edu

Abstract

The ability to reason about action and change has long been considered a necessary component for any intelligent system. Many proposals have been offered in the past to deal with this problem. In this paper, we offer a new approach to belief change associated with performing actions that addresses some of the shortcomings of these approaches. In particular, our approach is based on a well-developed theory of action in the situation calculus extended to deal with belief. Moreover, our account handles nested belief, belief introspection, mistaken belief, and handles belief revision and belief update together with iterated belief change.

1 Introduction

An agent acting in its environment must be capable of reasoning about the state of its environment and keeping track of any changes to the environment due to the performing of actions. Various theories have been developed to give an account of how this can be achieved. Foremost among these are theories of belief change and theories for reasoning about action. While originating from different initial motivations, the two are united in their aim to have agents maintain a model of the environment that matches the actual environment as closely as possible given the available information. An important consideration is the ability to deal with more than one change; known as the problem of *iterated belief change.*

In this paper, we consider a new approach for modeling iterated belief change using the language of the *situation calculus* [15]. While our approach is limited in its applicability, we feel that it is conceptually very simple and offers a number of useful features not found in other approaches:

- It is completely integrated with a well-developed theory of action in the situation calculus [18] and its extension to handle knowledge expansion [19]. Specifically, how beliefs change in our account is simply a special case of how other fluents change as the result of actions, and thus among other things, we inherit a solution to the frame problem.
- Like Scherl and Levesque [19], our theory accommodates both belief *update* and belief *expansion*. The former concerns beliefs that change as the result of the realization that the world has changed; the latter concerns beliefs that change as the result of new information acquired.
- Unlike Scherl and Levesque, however, our theory is not limited to belief expansion; rather it deals with the more general case of belief *revision*. It will be possible in our model for an agent to believe some formula ϕ, acquire information that causes it to change its mind and believe $\neg\phi$ (without believing the world has changed), and later go back to believing ϕ again. In Scherl and Levesque and in other approaches based on this work such as [12, 13], new information that contradicts previous beliefs cannot be consistently accommodated.
- Because belief change in our model is always the result of action, our account naturally supports *iterated* belief change. This is simply the result of a sequence of actions. Moreover, each individual action can potentially cause both an update (by changing the world) and a revision (by providing sensing information) in a seamless way.
- Like Scherl and Levesque and unlike many previous approaches to belief change, e.g., [9, 11], our approach supports belief *introspection*: an agent will know what it believes and does not believe. Furthermore, it has information about the past, and so will also know what it used to believe and not believe. Finally, an agent will be able to predict what it will believe after it acquires information through sensing.

- Unlike Scherl and Levesque, our agents will be able to introspectively tell the difference between an update and a revision as it moves from believing ϕ to believing $\neg\phi$. In the former case, the agent will believe that it believed ϕ in the past, and that it was correct to do so; in the latter case, it will believe that it believed ϕ in the past but that it was *mistaken*.

The rest of the paper is organized as follows: in the next section, we briefly review the situation calculus including the Scherl and Levesque [19] model of belief expansion, and we review the most popular accounts of belief revision, belief update and iterated belief change; in Section 3, we motivate and define a new belief operator as a modification to the one used by Scherl and Levesque; in Section 4, we prove some properties of this operator, justifying the points made above; in Section 5, we show the operator in action on a simple example, and how an agent can change its mind repeatedly; in Section 6, we consider the importance of our work and compare it to some of the existing approaches to belief change; in the final section, we draw some conclusions and discuss future work.

2 Background

The basis of our framework for belief change is an action theory [18] based on the situation calculus [15], and extended to include a belief operator [19]. In this section, we begin with a brief overview of the situation calculus and follow it with a short review of belief change.

2.1 Situation Calculus

The situation calculus is a predicate calculus language for representing dynamically changing domains. A situation represents a snapshot of the domain. There is a set of initial situations corresponding to the ways the agent[1] believes the domain might be initially. The actual initial state of the domain is represented by the distinguished initial situation constant, S_0, which may or may not be among the set of initial situations believed possible by the agent. The term $do(a, s)$ denotes the unique situation that results from the agent performing action a in situation s. Thus, the situations can be structured into a set of trees, where the root of each tree is an initial situation and the arcs are actions. The initial situations are defined as those situations that do not have a predecessor:

Definition 1

$$Init(s) \stackrel{\text{def}}{=} \neg\exists a, s'.s = do(a, s').$$

Predicates and functions whose value may change from situation to situation (and whose last argument is a situation) are called *fluents*. For instance, we use the fluent $\text{INR}_1(s)$ to represent that the agent is in room R_1 in situation s. The effects of actions on fluents are defined using successor state axioms [18], which provide a succinct representation for both effect axioms and frame axioms [15]. For example, assume that there are only two rooms, R_1 and R_2, and that the action LEAVE takes the agent from the current room to the other room. Then, the successor state axiom for INR_1 is:[2]

$$\text{INR}_1(do(a, s)) \equiv$$
$$((\neg\text{INR}_1(s) \wedge a = \text{LEAVE}) \vee (\text{INR}_1(s) \wedge a \neq \text{LEAVE})).$$

This axiom asserts that the agent will be in R_1 after doing some action iff either the agent is in R_2 ($\neg\text{INR}_1(s)$) and leaves it or the agent is currently in R_1 and the action is anything other than leaving it.

Moore [16] defined a possible-worlds semantics for a modal logic of knowledge in the situation calculus by treating situations as possible worlds. Scherl and Levesque [19] adapted the semantics to the action theories of Reiter [18]. The idea is to have an accessibility relation on situations, $B(s', s)$, which holds if in situation s, the situation s' is considered possible by the agent. Note, the order of the arguments is reversed from the usual convention in modal logic.

Levesque [13] introduced a predicate, $SF(a, s)$, to describe the result of performing the binary-valued sensing action a. $SF(a, s)$ holds iff the sensor associated with a returns the sensing value 1 in situation s. Each sensing action senses some property of the domain. The property sensed by an action is associated with the action using a *guarded sensed fluent axiom* [10]. For example, suppose that there are lights in R_1 and R_2 and that $\text{LIGHT}_1(s)$ ($\text{LIGHT}_2(s)$, resp.) holds if the light in R_1 (R_2, resp.) is on. Then:

$$\text{INR}_1(s) \supset (SF(\text{SENSELIGHT}, s) \equiv \text{LIGHT}_1(s))$$
$$\neg\text{INR}_1(s) \supset (SF(\text{SENSELIGHT}, s) \equiv \text{LIGHT}_2(s))$$

can be used to specify that the SENSELIGHT action senses whether the light in the room where the agent is currently located is on.

Scherl and Levesque [19] defined a successor state axiom for B that shows how actions, including sensing actions, affect the beliefs of the agent. We use the same axiom (with some notational variation) here:

Axiom 1 *(Successor State Axiom for B)*

$$B(s'', do(a, s)) \equiv$$
$$\exists s'[B(s', s) \wedge s'' = do(a, s') \wedge (SF(a, s') \equiv SF(a, s))].$$

The situations s'' that are B-related to $do(a, s)$ are the ones that result from doing action a in a situation s', such that the sensor associated with action a has the same value in s' as it does in s. We will see in Section 3 how a modal belief operator can be defined in terms of this fluent.

[1] The situation calculus can accommodate multiple agents, but for the purposes of this paper we assume that there is a single agent, and all actions are performed by that agent.

[2] We adopt the convention that unbound variables are universally quantified in the widest scope.

There are various ways of axiomatizing dynamic applications in the situation calculus. Here we adopt a simple form of the guarded action theories described by De Giacomo and Levesque [10] consisting of: (1) successor state axioms[3] for each fluent (including B and pl introduced below), and guarded sensed fluent axioms for each action, as discussed above; (2) unique names axioms for the actions, and domain-independent foundational axioms (similar to the ones given by Lakemeyer and Levesque [12]), which we do not describe further here; and (3) initial state axioms, which describe the initial state of the domain and the initial beliefs of the agent.[4] For simplicity, we assume here that all actions are always executable and omit the action precondition axioms and references to a *Poss* predicate that are normally included in situation calculus action theories.

In what follows, we will use Σ to refer to a guarded action theory of this form. By a *domain-dependent fluent*, we mean a fluent other than B or pl, and a *domain-dependent formula* is one that only mentions domain-dependent fluents. Finally, we say that a domain-dependent formula is *uniform* in s iff s is the only situation term in that formula.

2.2 Belief Change

Before formally defining a belief operator in this language, we briefly review the notion of belief change as it exists in the literature. Belief change, simply put, aims to study the manner in which an agent's epistemic (belief) state should change when the agent is confronted by new information. In the literature,[5] there is often a clear distinction between two forms of belief change: *revision* and *update*. Both forms can be characterized by an axiomatic approach (in terms of rationality postulates) or through various constructions (e.g., epistemic entrenchment, possible worlds, etc.). The AGM theory [9] is the prototypical example of belief revision while the KM framework [11] is often identified with belief update.

Intuitively speaking, belief revision is appropriate for modeling static environments about which the agent has only partial and possibly incorrect information. New information is used to fill in gaps and correct errors, but the environment itself does not undergo change. Belief update, on the other hand, is intended for situations in which the environment itself is changing due to the performing of actions.

For completeness and later comparison, we list here the AGM postulates [1, 9] for belief revision. By $K*\phi$ we mean the revision of belief state K by new information ϕ.[6]

(K*1) $K * \phi$ is deductively closed
(K*2) $\phi \in K * \phi$
(K*3) $K * \phi \subseteq K + \phi$
(K*4) If $\neg\phi \notin K$, then $K + \phi \subseteq K * \phi$
(K*5) $K * \phi = \mathcal{L}$ iff $\models \neg\phi$
(K*6) If $\models \phi \equiv \psi$, then $K * \phi = K * \psi$
(K*7) $K * (\phi \wedge \psi) \subseteq (K * \phi) + \psi$
(K*8) If $\neg\psi \notin K * \phi$, then $(K * \phi) + \psi \subseteq K * (\phi \wedge \psi)$

Katsuno and Mendelzon provide the following postulates for belief update, where $K \diamond \phi$ denotes the update of K by formula ϕ.[7]

(K⋄1) $K \diamond \phi$ is deductively closed
(K⋄2) $\phi \in K \diamond \phi$
(K⋄3) If $\phi \in K$, then $K \diamond \phi = K$
(K⋄4) $K \diamond \phi = \mathcal{L}$ iff $K \models \bot$ or $\phi \models \bot$
(K⋄5) If $\models \phi \equiv \psi$, then $K \diamond \phi = K \diamond \psi$
(K⋄6) $K \diamond (\phi \wedge \psi) \subseteq (K \diamond \phi) + \psi$
(K⋄7) If K is complete and $\neg\psi \notin K \diamond \phi$, then $(K \diamond \phi) + \psi \subseteq K \diamond (\phi \wedge \psi)$
(K⋄8) If $[K] \neq \emptyset$, then $K \diamond \phi = \bigcap_{w \in [K]} w \diamond \phi$

One of the major issues in this area is that of *iterated belief change*, i.e., modeling how the agent's beliefs change after multiple belief revisions or updates occur. Two of the main developments in this area are the work of Darwiche and Pearl [6] and Boutilier [4]. Darwiche & Pearl put forward the following postulates as a way of extending the AGM revision postulates to handle *iterated revision*.[8]

(DP1) If $\psi \models \phi$, then $(K * \phi) * \psi = K * \psi$
(DP2) If $\psi \models \neg\phi$, then $(K * \phi) * \psi = K * \psi$
(DP3) If $\phi \in K * \psi$, then $\phi \in (K * \phi) * \psi$
(DP4) If $\neg\phi \notin K * \psi$, then $\neg\phi \notin (K * \phi) * \psi$

In Section 6.2, we return to consider the extent to which our framework satisfies these postulates.

[3] We could use the more general *guarded successor state axioms* of De Giacomo and Levesque [10], but regular successor state axioms suffice for the simple domain we consider here.

[4] These are axioms that only describe initial situations. Reiter [18] has S_0 as the only initial situation, but to formalize belief, we need additional ones.

[5] We shall restrict our attention to approaches in the AGM vein [1, 9, 11] although there are many others.

[6] In the AGM theory, K is a set of formulae and ϕ is a formula taken from an object language $\mathcal{L}$ containing the standard boolean connectives and the logical constant $\bot$ (falsum). Furthermore, K is a set of formulae (from $\mathcal{L}$) closed under the deductive consequence operator Cn associated with the underlying logic. The operation $K + \phi$ denotes the belief expansion of K by ϕ and is defined as $K + \phi = Cn(K \cup \{\phi\})$. $[K]$ denotes the set of all consistent complete theories of $\mathcal{L}$ containing K.

[7] To facilitate comparison with the AGM postulates, we have reformulated the original postulates of Katsuno and Mendelzon into an equivalent set using AGM-style terminology [17]. For renderings of these postulates and the AGM postulates above in the KM-style, refer to Katsuno & Mendelzon [11].

[8] Again, we have translated the Darwiche and Pearl postulates into AGM-style terminology rather than KM-style terminology used in the original paper.

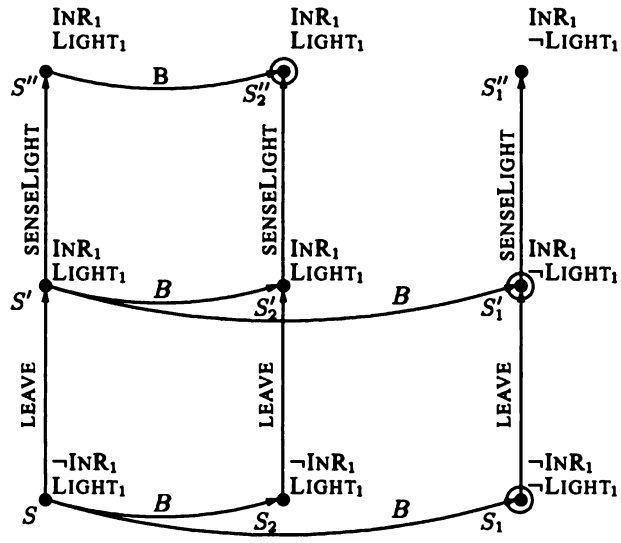


Figure 1: An example of belief update and revision.

3 Definition of the Belief Operator

In this section, we define what it means for an agent to believe a formula ϕ in a situation s, i.e., $Bel(\phi, s)$. Since ϕ will usually contain fluents, we introduce a special symbol *now* as a placeholder for the situation argument of these fluents, e.g., $Bel(\text{INR}_1(now), s)$. $\phi[s]$ denotes the formula that results from substituting s for *now* in ϕ. To make the formulae easier to read, we will often suppress the situation argument of fluents in the scope of a belief operator, e.g., $Bel(\text{INR}_1, s)$.

Scherl and Levesque [19], define a modal operator for belief in terms of the accessibility relation on situations, $B(s', s)$. For Scherl and Levesque, the believed formulae are the ones true in all accessible situations:

Definition 2

$$Bel_{\text{SL}}(\phi, s) \stackrel{\text{def}}{=} \forall s'(B(s', s) \supset \phi[s']).$$

To understand how belief change works, both in Scherl and Levesque and here, consider the example illustrated in Figure 1. In this example, we have three initial situations S, S_1, and S_2. S_1 and S_2 are B-related to S (i.e., $B(S_1, S)$ and $B(S_2, S)$), as indicated by the arrows labeled B. (Ignore the circles around certain situations for now.) In all three situations, the agent is not in the room R_1. In S and S_2 the light in R_1 is on, and in S_1 the light is off. So at S, the agent believes it is not in R_1 (i.e., that it is in R_2), but it has no beliefs about the status of the light in R_1. We first consider the action of leaving R_2, which will lead to a belief update. By the successor state axiom for B, both $do(\text{LEAVE}, S_1)$ and $do(\text{LEAVE}, S_2)$ are B-related to $do(\text{LEAVE}, S)$. In the figure, these three situations are called S'_1, S'_2 and S', respectively. The successor state axiom for INR_1 causes INR_1 to hold in these situations. Therefore, the agent believes INR_1 in S'. By the successor state axiom for LIGHT_1, which we state below, the truth value of LIGHT_1 would not change as the result of LEAVE.

Now the agent performs the sensing action SENSELIGHT. According to the sensed fluent axioms for SENSELIGHT, $SF(\text{SENSELIGHT}, S^*)$ holds for situation S^* iff the light is on in the room in which the agent is located in S^*. In the figure, the light in R_1 is on in S' and S'_2, but not in S'_1. So, SF holds for SENSELIGHT in the former two situations but not in the latter. The successor state axiom for B ensures that after doing a sensing action A, any situation that disagrees with the actual situation on the value of SF for A is dropped from the B relation in the successor state. In the figure, S' is the actual situation. Since S'_1 disagrees with S' on the value of SF for SENSELIGHT, $do(\text{SENSELIGHT}, S'_1)$ (labeled S''_1 in the figure) *is not* B-related to $do(\text{SENSELIGHT}, S')$ (labeled S''). On the other hand, S'_2 and S' agree on the value of SF for SENSELIGHT, so $do(\text{SENSELIGHT}, S'_2)$ (labeled S''_2 in the figure) *is* B-related to S''. The result is that the agent believes the light is on in S''. This is an example of belief expansion because the belief that the light is on was simply added to the belief state of the agent.

Our definition of *Bel* is similar to the one in Scherl and Levesque, but we are going to generalize their account in order to be able to talk about how *plausible* the agent considers a situation to be. Plausibility is assigned to situations using a function $pl(s)$, whose range is the natural numbers, where lower values indicate higher plausibility. The pl function only has to be specified over initial situations, using an initial state axiom. The plausibility of successor situations is left unchanged using the following successor state axiom:

Axiom 2 *(Successor State Axiom for pl)*

$$pl(do(a, s)) = pl(s).$$

Unlike Scherl and Levesque, we will say that the agent believes a proposition ϕ in situation s, if ϕ holds in the *most plausible* B-related situations. Here is our definition of the belief operator:

Definition 3

$$Bel(\phi, s) \stackrel{\text{def}}{=} \forall s'[B(s', s) \wedge (\forall s''.B(s'', s) \supset pl(s') \leq pl(s''))] \supset \phi[s'].$$

That is, ϕ is believed at s precisely when it holds at all the most plausible situations B-related to s. Note that the actual numbers assigned to the situations are not relevant. All that is important is the ordering of the situations by plausibility. We could have used any total pre-order on situations for this purpose, but using $\leq$ on natural numbers simplifies the presentation of our framework.

We now return to the initial situations in Figure 1, and add a plausibility structure to the belief state of the agent by supposing that S_1 is more plausible than S_2 (indicated by the circle surrounding S_1). For example, suppose that $pl(S_1) = 0$ and $pl(S_2) = 1$. Now, the beliefs of the agent are determined only by S_1. Therefore, the agent now has a belief about the light in R_1 in S, namely that the light is off. After leaving R_2 and entering R_1, the agent continues to believe that the light is off. After doing SENSELIGHT, S_1'' is dropped from B as before, so now S_2'' is the most plausible accessible situation, which means that it determines the beliefs of the agent. Since the light is on in S_2'', the agent believes it is on in S''. Since the agent goes from believing the light is off to believing it is on, this is a case of belief revision.

Both accounts of belief handle *belief introspection* of current and past beliefs. In order to obtain positive and negative introspection of beliefs, we require B to be initially transitive and euclidean. For notational simplicity, we combine the two constraints into a single constraint, which says that any situation that is B-related to an initial situation s is B-related to the same situations as s. We assert this constraint as an initial state axiom:

Axiom 3

$$Init(s) \wedge B(s', s) \supset (\forall s''.B(s'', s') \equiv B(s'', s)).$$

As in Scherl and Levesque, the successor state axiom for B ensures that this constraint is preserved over all situations:

Theorem 1

$$B(s', s) \supset (\forall s''.B(s'', s') \equiv B(s'', s)).$$

In order to clarify how this constraint ensures that introspection is handled properly, we will show that in the example illustrated in Figure 1, the agent positively introspects its past beliefs. First, we need some notation that allows us to talk about the past. We use $Previously(\phi, s)$ to denote that ϕ held in the situation immediately before s:

Definition 4

$$Previously(\phi, s) \stackrel{\text{def}}{=} \exists a, s'.s = do(a, s') \wedge \phi[s'].$$

We want to show that $Bel(Previously(Bel(\neg \text{LIGHT}_1)), S'')$[9] holds, i.e., in S'', the agent believes that in the previous situation it believed that the light in R_1 was off. Consider a situation S^* that is among the most plausible B-related situations to S''. In this example, there is only one such situation, namely, S_2''. We need to show that $Previously(Bel(\neg \text{LIGHT}_1), S_2'')$ holds, i.e., that $Bel(\neg \text{LIGHT}_1, S_2')$ holds. By Theorem 1, S_2' is B-related to the same situations as S', i.e., S_1' and S_2'. Since S_1' is more plausible than S_2', we only require that $\neg \text{LIGHT}_1(S_1')$ holds. Since this is true, we see that $Bel(Previously(Bel(\neg \text{LIGHT}_1)), S'')$ is also true.

The specification of pl and B over the initial situations is the responsibility of the axiomatizer of the domain in question. This specification need not be complete. Of course, a more complete specification will yield more interesting properties about the agent's current and future belief states.

We have another constraint on the specification of B over the initial situations: the situations B-related to an initial situation are themselves initial, i.e., the agent believes that initially nothing has happened. We assert this constraint as an initial state axiom:

Axiom 4

$$Init(s) \wedge B(s', s) \supset Init(s').$$

4 Properties

In this section, we highlight some of the more interesting properties of our framework. In order to clarify our explanations and facilitate a comparison with previous approaches to belief change, it will be important for us to attach a specific meaning to the use of the terms *revision* and *update*, which we shall do here.

4.1 Belief Revision

Recall (Section 2.2) that belief revision is suited to the acquisition of information about static environments for which the agent may have mistaken or partial information. In our framework, this can only be achieved through the use of sensing actions. We suppose that to revise by a formula ϕ, there is a corresponding sensing action capable of determining the truth value of ϕ. Moreover, we assume that this sensing action has no effect on the environment; the only fluent it changes is B.[10]

More formally, we define a revision action as follows:

Definition 5 *(Revision Action for ϕ)*
A revision action A for a formula ϕ (uniform in now) wrt action theory Σ is a sensing action satisfying $\Sigma \models [\forall s.SF(A, s) \Leftrightarrow \phi[s]] \wedge [\forall s \forall \vec{x}.F(\vec{x}, s) \Leftrightarrow F(\vec{x}, do(A, s))]$ (for every domain-dependent fluent F).

We now show that belief revisions are handled appropriately in our system in the sense that if the sensor indicates that ϕ holds, then the agent will indeed believe ϕ after performing A. Similarly, if the sensor indicates that ϕ is false, then the agent will believe $\neg\phi$ after doing A.

Theorem 2
Let A be a revision action for formula ϕ (uniform in now).

[9] Recall that we omit the situation argument of fluents in the scope of a *Bel* operator whenever possible.

[10] This is not an overly strict imposition for we can capture sensing actions that modify the domain by "decomposing" the action into a sequence of non-sensing actions and sensing actions.

It follows that:

$$\Sigma \models [\forall s.\phi[s] \supset Bel(\phi, do(A,s))] \wedge [\forall s.\neg\phi[s] \supset Bel(\neg\phi, do(A,s))]$$

If the agent is indifferent towards ϕ before doing the action, i.e., does not believe ϕ or $\neg\phi$, this is a case of belief expansion. If, before sensing, the agent believes the opposite of what the sensor indicates, then we have belief revision.

Note that this theorem also follows from Scherl and Levesque's theory. However, for Scherl and Levesque, if the agent believes ϕ in S and the sensor indicates that ϕ is false, then in $do(A,S)$, the agent's belief state will be inconsistent. The agent will then believe all propositions, including $\neg\phi$. In our theory, the agent's belief state will be consistent in this case, as long as there is some situation S', accessible from S that agrees with S on the value of the sensor associated with A (here, A can be any action, not just a revision action):

Theorem 3

$$\Sigma \models \forall a,s\{[\exists s'.B(s',s) \wedge (SF(a,s') \equiv SF(a,s))] \supset \neg Bel(FALSE, do(a,s))\}$$

Since S' is not necessarily among the *most plausible* accessible situations, the agent can consistently believe ϕ in S and $\neg\phi$ in $do(A,S)$. As a direct corollary to this result, if we restrict our attention to revision actions for ϕ where the agent considers ϕ is possible, it will not hold inconsistent beliefs after performing A.

Corollary 4

Let A be a revision action for a formula ϕ (uniform in now). It follows that:

$$\Sigma \models (\exists s'.B(s',s) \wedge (\phi[s'] \equiv \phi[s]) \supset \neg Bel(FALSE, do(A,s)).$$

4.2 Belief Update

Belief update refers to the belief change that takes place due to a change in the environment. In analogy to revision, we introduce the notion of an update action. (Recall that we assume that actions are always possible.)

Definition 6 *(Update Action for ϕ)*

An update action A for a formula ϕ (uniform in now) wrt action theory Σ is a non-sensing action that always makes ϕ true in the environment. That is, $\Sigma \models \forall s.\phi[do(A,s)] \wedge \forall s.SF(A,s)$.

As with Scherl and Levesque's theory, the agent's beliefs are updated appropriately when an update action A for ϕ occurs, in the sense that the agent will believe ϕ after A is performed.

Theorem 5

Let A be an update action for a formula ϕ (uniform in now). It follows that:

$$\Sigma \models \forall s.\ Bel(\phi,\ do(A,\ s))$$

In our framework, we can represent actions that do not fall under the category of update actions. Of particular interest are ones whose effects depend on what is true in the current situation. We can prove an analogous theorem for such actions. Let A be a non-sensing action, i.e., $\forall s.SF(A,s)$. Further suppose that A is an action that causes ϕ' to hold, if ϕ holds beforehand, and that the agent believes ϕ in S. Then after performing A in S, the agent ought to believe that ϕ' holds:

Theorem 6

$$\Sigma \models Bel(\phi,s) \wedge (\forall s'.SF(A,s')) \wedge (\forall s'.\phi[s'] \supset \phi'[do(A,s')]) \supset Bel(\phi', do(A,s)).$$

4.3 Introspection

In Section 3, we claimed that the agent can introspect its beliefs. We do indeed have this as a theorem.

Theorem 7

$$\Sigma \models [Bel(\phi,s) \supset Bel(Bel(\phi),s)] \wedge [\neg Bel(\phi,s) \supset Bel(\neg Bel(\phi),s)].$$

This is a straightforward consequence of Theorem 1.

4.4 Awareness of Mistakes

In Section 3, we also claimed that the agent can introspect its past beliefs. Now suppose that the agent believes ϕ in S, and after performing a sensing action A in S, the agent discovers that ϕ is false. In $do(A,S)$, the agent should believe that in the previous situation ϕ was false, but it believed ϕ was true. In other words, the agent should believe that it was mistaken about ϕ. We now state a theorem that says that the agent will indeed believe that it was mistaken about ϕ. First note that this only holds if A does not affect ϕ. If A causes ϕ to become false, then there is no reason for the agent to believe that ϕ was false in the previous situation. In the theorem, we rule out this case by stating in the antecedent that for any situation S', ϕ holds in S' iff ϕ holds in $do(A,S')$.

Theorem 8

$$\Sigma \models Bel(\phi,s) \wedge Bel(\neg\phi, do(a,s)) \wedge (\forall s'.\phi[s'] \equiv \phi[do(a,s')]) \supset Bel(Previously(\neg\phi \wedge Bel(\phi)), do(a,s)).$$

In Section 6.2, we will discuss to what extent standard AGM revision and KM update postulates are satisfied in our framework.

5 Example

We now present an example to illustrate how this theory of belief change can be applied. We model a world in which there are two rooms, R_1 and R_2. The agent can move between the rooms. Each room contains a light that can be on

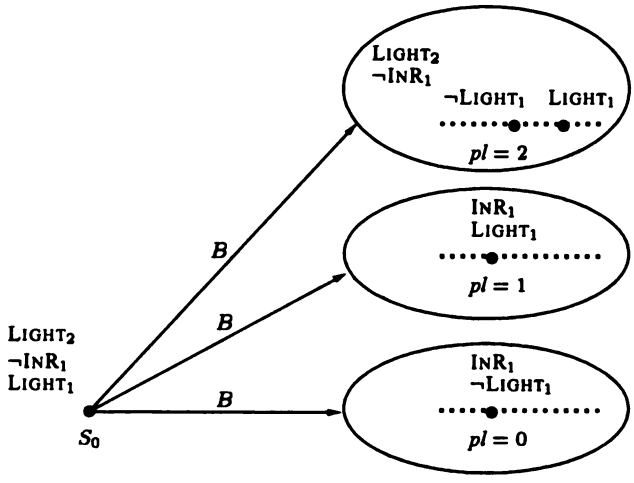


Figure 2: The initial state of the example domain.

or off. The agent has two binary sensors. One sensor detects whether or not the light is on in the room in which the agent is currently located. The other sensor detects whether or not the agent is in R_1.

We have three fluents: $\text{LIGHT}_1(s)$ ($\text{LIGHT}_2(s)$, resp.), which holds iff there is light in R_1 (R_2, resp.) in situation s, and $\text{INR}_1(s)$, which holds if the agent is in R_1 in s. If the agent is not in R_1, then it is assumed to be in R_2. There are three actions: the agent leaves the room it is in and enters the other room (LEAVE), the agent senses whether it is in R_1 (SENSEINR_1), and the agent senses whether the light is on in the room in which it is currently located (SENSELIGHT).

The successor state axioms and guarded sensed fluent axioms for our example, which we will call E, are as follows:

$$\text{LIGHT}_1(do(a,s)) \equiv \text{LIGHT}_1(s)$$
$$\text{LIGHT}_2(do(a,s)) \equiv \text{LIGHT}_2(s)$$
$$\text{INR}_1(do(a,s)) \equiv$$
$$((\neg\text{INR}_1(s) \wedge a = \text{LEAVE}) \vee (\text{INR}_1(s) \wedge a \neq \text{LEAVE}))$$
$$\text{TRUE} \supset (SF(\text{LEAVE}, s) \equiv \text{TRUE})$$
$$\text{INR}_1(s) \supset (SF(\text{SENSELIGHT}, s) \equiv \text{LIGHT}_1(s))$$
$$\neg\text{INR}_1(s) \supset (SF(\text{SENSELIGHT}, s) \equiv \text{LIGHT}_2(s))$$
$$\text{TRUE} \supset (SF(\text{SENSEINR}_1, s) \equiv \text{INR}_1(s))$$

Next we must specify the initial state. This includes both the physical state of the domain and the belief state of the agent. First we describe the initial physical state of the domain, by saying which domain-dependent fluents hold in the actual initial situation, S_0. Initially, the lights in both rooms are on and the agent is in R_2 (this is illustrated on the left side of Figure 2):

$$\text{LIGHT}_1(S_0) \wedge \neg\text{INR}_1(S_0) \wedge \text{LIGHT}_2(S_0).$$

The initial belief state of the agent is illustrated in Figure 2. It shows that in the most plausible (the ones with plausibility 0 in the figure) B-related situations to S_0, $\neg\text{LIGHT}_1$ and INR_1 hold. In the next most plausible (the ones with plausibility 1) B-related situations to S_0, LIGHT_1 and INR_1 hold. In the third most plausible (the ones with plausibility 2) B-related situations to S_0, LIGHT_2 and $\neg\text{INR}_1$ hold. There is also at least one situation in the latter group in which LIGHT_1 holds and one in which $\neg\text{LIGHT}_1$ holds. Specifying this belief state directly can be cumbersome. For example, the axiom for the situations with plausibility 1 is:

$$(\exists s.Init(s) \wedge B(s, S_0) \wedge pl(s) = 1) \wedge$$
$$(\forall s.Init(s) \wedge pl(s) = 1 \supset \text{LIGHT}_1(s) \wedge \text{INR}_1(s)).$$

For now, we will not enumerate the set of axioms that specify the belief state shown in Figure 2. But we assume that we have such a set which, together with the axioms for the initial physical state, we refer to as I. After we have discussed the example, we will show that there is a more elegant way to specify the initial belief state of the agent. So for this example, Σ consists of the foundational axioms, unique names axioms, Axioms 1–4, E, and I, for which we get the following:

Theorem 9 *The following formulae are entailed by* Σ*:*

i. $Bel(\neg\text{LIGHT}_1 \wedge \text{INR}_1, S_0)$

ii. $Bel(\text{LIGHT}_1 \wedge \text{INR}_1, do(\text{SENSELIGHT}, S_0))$

iii. $Bel(\neg\text{INR}_1, do(\text{SENSEINR}_1, do(\text{SENSELIGHT}, S_0)))$

iv. $Bel(Previously(\neg\text{INR}_1 \wedge Bel(\text{INR}_1)),$
$do(\text{SENSEINR}_1, do(\text{SENSELIGHT}, S_0)))$

v. $\neg Bel(\text{LIGHT}_1, do(\text{SENSEINR}_1, do(\text{SENSELIGHT}, S_0))) \wedge$
$\neg Bel(\neg\text{LIGHT}_1, do(\text{SENSEINR}_1, do(\text{SENSELIGHT}, S_0)))$

vi. $Bel(\text{INR}_1,$
$do(\text{LEAVE}, do(\text{SENSEINR}_1, do(\text{SENSELIGHT}, S_0))))$

vii. $Bel(\text{LIGHT}_1,$
$do(\text{SENSELIGHT},$
$do(\text{LEAVE}, do(\text{SENSEINR}_1,$
$do(\text{SENSELIGHT}, S_0)))))$.

We shall now give a short, informal explanation of why each part of the previous theorem holds.

i. In the most plausible situations B-related to S_0, $\neg\text{LIGHT}_1 \wedge \text{INR}_1$ holds.

ii. Even though the agent believes that it is in R_1 initially, it is actually in R_2. Therefore, its light sensor is measuring whether there is light in R_2, even though the agent thinks that it is measuring whether there is light in R_1. It turns out that there is light in R_2 in S_0, so the sensor returns 1. Since the agent believes that the light sensor is measuring whether there is light in R_1 and in all the situations with plausibility 0, there is no light in R_1, those situations are dropped from the B relation. In the situations with plausibility 1, the light is on in R_1, so those situations are retained. In those situations $\text{LIGHT}_1 \wedge \text{INR}_1$ holds and those fluents are not affected by the SENSELIGHT action, so the agent believes $\text{LIGHT}_1 \wedge \text{INR}_1$ after doing SENSELIGHT.

iii. Now the agent senses whether it is in R_1. Again the agent's most plausible situations conflict with what is actually the case, so they are dropped from the B relation. The situations with plausibility 2 become the most plausible situations, so the agent believes it is not in R_1.

iv. By Theorem 8, the agent realizes that it was mistaken about being in R_1.

v. Among the situations with plausibility 2, there is one in which the light is on in R_1 and one in which it is not on. Therefore, the agent is unsure as to whether the light is on.

vi. Now the agent leaves R_2 and enters R_1. This happens in all the B-related situations as well. Therefore, the agent believes that it is in R_1. This is an example of an update.

vii. The light in R_1 was on initially, and since no action was performed that changed the state of the light, the light remains on. After checking its light sensor, the agent believes that the light is on in R_1.

This example shows that the agent's beliefs change appropriately after both revision actions and update actions. The example also demonstrates that our formalism can accommodate iterated belief change. The agent goes from believing that the light is not on, to believing that it is on, to not believing one way or the other, and then back to believing that it is on.

To facilitate the specification of the initial belief state of the agent, we find it convenient to define another belief operator $\Rightarrow$, in the spirit of the conditional logic connective [14]:

Definition 7

$$\phi \Rightarrow_s \psi \stackrel{\text{def}}{=} \forall s'[B(s', s) \wedge \phi[s'] \wedge \forall s''(B(s'', s) \wedge \phi[s''] \supset pl(s') \leq pl(s'')) \supset \psi[s']].$$

$\phi \Rightarrow_s \psi$ holds if in the most plausible situations B-related to s where ϕ holds, ψ also holds. Note that for any situation S, $Bel(\phi, S)$ is equivalent to $(\text{TRUE} \Rightarrow_S \phi)$.

We can use this operator to specify the initial belief state of the agent without having to explicitly mention the plausibility of situations. To obtain the results of Theorem 9, it suffices to let I be the following set of axioms:

$$\begin{aligned}
&\text{LIGHT}_1(S_0) \wedge \neg\text{IN}R_1(S_0) \wedge \text{LIGHT}_2(S_0)\\
&\text{TRUE} \Rightarrow_{S_0} \neg\text{LIGHT}_1 \wedge \text{IN}R_1\\
&\text{LIGHT}_1 \Rightarrow_{S_0} \text{IN}R_1\\
&\neg(\text{LIGHT}_2 \wedge \neg\text{IN}R_1 \Rightarrow_{S_0} \text{LIGHT}_1)\\
&\neg(\text{LIGHT}_2 \wedge \neg\text{IN}R_1 \Rightarrow_{S_0} \neg\text{LIGHT}_1)
\end{aligned}$$

It is easy to see that the belief state depicted in Figure 2 satisfies these axioms. In the most plausible worlds, $(\neg\text{LIGHT}_1 \wedge \text{IN}R_1)$ holds. In the most plausible worlds where the light in R_1 is on, the agent is in R_1. Finally, the last two axioms state that among the most plausible worlds where the light is on in R_2 and the agent is in R_2, there is one where the is light is off in R_1 and one in which the light is on (resp.).

6 Discussion

There are various aspects of our framework that deserve further consideration. We address what we consider to be some of the more important issues here.

6.1 Plausibility Ordering

Our plausibility function is based on ordinal conditional (κ) functions [6, 21]. However, our assignment of plausibilities to situations is fixed, whereas the plausibility assigned to a world using a κ-function can change when revisions occur. The dynamics of belief in our framework derives from the dynamics of the B relation, rather than that of the plausibility assignment.

In Darwiche and Pearl's framework [6], the κ-ranking of a world that does not satisfy the formula in a revision increases by 1. However, if the world satisfies the revision formula in future revisions, the world's κ-ranking decreases, and if it decreases to 0, the world will help determine the beliefs of the agent. In our framework, when a sensing action occurs, any situation S' that disagrees with the actual value of the sensor is *removed* from the B relation (actually, its successor is removed). The successors of S' will never be readmitted to B, so they will never help determine the beliefs of the agent.

One may think that having a fixed plausibility assignment limits the applicability of our approach. Consider an example[11] where, most plausibly, a cat is asleep at home, but where after phoning home, most plausibly, the cat is awake. (Nothing is certain in either case.) This might seem to require adjustment of the plausibility assignment.

To handle this example, we need first to observe that in the action theory we are using, actions are taken to be *deterministic*, with effects described by successor state axioms, quite apart from properties of belief and plausibility. If in some situations a phone action wakes the cat, and in others not, then there has to be some property M such that we can write a successor state axiom of the following form:

$$\begin{aligned}
&\text{AWAKE}(do(a, s)) \equiv (a = \text{PHONE} \wedge M(s))\\
&\quad \vee [\ldots\text{other actions that can wake cats}\ldots]\\
&\quad \vee (\text{AWAKE}(s) \wedge [a \text{ is not some put-to-sleep action}]).
\end{aligned}$$

For example, M could represent that "the phone's ringer is loud enough to wake the cat". With this model, we can then arrange the B relation in the initial situation so that there are 4 groups of situations s' B-related to S_0 where

[11] We are indebted to Jim Delgrande for this example.

the following hold (in order of decreasing plausibility): $M(s') \wedge \neg\text{AWAKE}(s')$, $M(s') \wedge \text{AWAKE}(s')$, $\neg M(s') \wedge \neg\text{AWAKE}(s')$, and $\neg M(s') \wedge \text{AWAKE}(s')$. Then we obtain:

$$Bel(\neg\text{AWAKE}, S_0)$$

but

$$Bel(\text{AWAKE}, do(\text{PHONE}, S_0))$$

as desired.[12] Of course, we also get that

$$Bel(M, S_0)$$

but this is to be expected: why would we think it most likely that the cat would be awake after the phone rings if we didn't also think it most likely that the ringer was loud enough to waken it? Thus, changing our minds about the plausibility of the cat being awake does not require us to change the plausibility ordering over situations.

We can also handle a belief-revision variant of this example where we change our mind about whether phoning home wakes the cat. For example, imagine a sensing action EXAMINERINGER that informs us that M is false initially (*e.g.*, the ringer on the phone is set to low). Then, we get

$$Bel(\neg\text{AWAKE}, do(\text{PHONE}, do(\text{EXAMINERINGER}, S_0))).$$

In fact, in the process of developing the approach described in this paper, we experimented with various schemes where the plausibility assigned to situations could be updated. But we found that this led to problems for introspection. Consider a scheme where we combine the plausibility assignment with the belief accessibility relation by adding an extra argument to the B relation, i.e., where $B(s', n, s)$ means that in situation s the agent thinks s' is plausible to degree n. In order to ensure that beliefs are properly introspected, the relation would have to satisfy a constraint similar to the one given in Theorem 1, but taking plausibilities into account. That is to say, all the B-related situations to a situation s must have the same belief structure as s, i.e., they should be B-related to the same situations with the same plausibilities as s. Unfortunately, this conflicts with some of our intuitions about how to change plausibilities to accommodate new information.

Consider an example where we have two situations S_0 and S_1, and where initially the agent considers situation S_1 more plausible than S_0, i.e., $B(S_1, 0, S_0)$, $B(S_0, 1, S_0)$, $B(S_1, 0, S_1)$, $B(S_0, 1, S_1)$. Notice that S_0 and S_1 have the same belief structure. Suppose that $\text{LIGHT}_1(S_0) \wedge SF(\text{SENSELIGHT}, S_0)$ and $\neg\text{LIGHT}_1(S_1) \wedge \neg SF(\text{SENSELIGHT}, S_1)$ hold. The natural way to update the plausibilities after sensing would be to make the most plausible situations from a situation $do(\text{SENSELIGHT}, s)$ be the ones that agree with s on the value of $SF(\text{SENSELIGHT})$. So, if we let S_0' denote $do(\text{SENSELIGHT}, S_0)$ and S_1' denote $do(\text{SENSELIGHT}, S_1)$, then in S_0', S_0' should be more plausible than S_1' and in S_1', S_1' should be more plausible than S_0'. But this would violate the constraint that B-related situations have the same belief structure, and cause introspection to fail.

One way to avoid this problem would be to update the plausibilities of all situations based on what holds in the 'actual' situations, i.e., S_0 and its successors (this focuses attention on beliefs that hold in actual situations, which is what we normally do anyway). Friedman and Halpern [8] essentially use this approach. For the example above, we would look at how the plausibilities should change in S_0' and adjust the plausibilities in the situations B-related to S_0' (in this case just S_1') in the same way. We would then have that S_0' is more plausible than S_1' in both S_0' and S_1', i.e., $B(S_0', 0, S_0')$, $B(S_1', 1, S_0')$, $B(S_0', 0, S_1')$, $B(S_1', 1, S_1')$. Notice that S_0' and S_1' have the same belief structure, so the constraint violation mentioned above is resolved.

Unfortunately, under this new scheme we have a problem with beliefs about future beliefs. If we were to redefine *Bel* in the obvious way to accommodate the extra argument in B, our example would entail the very counterintuitive $Bel(\neg\text{LIGHT}_1 \wedge Bel(\text{LIGHT}_1, do(\text{SENSELIGHT}, now)), S_0)$, i.e., in S_0, the agent believes that the light is not on but thinks that after sensing he will believe that it is on. Our approach—which uses a fixed plausibility ordering on situations and simply drops situations that conflict with sensing results from the B relation—avoids both of these problems.

Another interesting difference between our approach and many of the proposals for modifying the plausibility ordering [4, 6, 21, 22] is that they adopt orderings over possible worlds which do not contain a history of the actions that have taken place in the world. Our approach, on the other hand, is based on situations, which do have such histories. While Friedman and Halpern [8] do not adopt situations, their possible worlds (runs) do include a history.

6.2 Comparison with AGM and KM

In order to effect a comparison with established belief change frameworks—in particular, the AGM and KM frameworks—we need to first establish a common footing. The first notion to establish is what is meant by the epistemic (or belief) state of the agent. We define a belief state (relative to a given situation) to consist of those formulae believed true at a particular situation. We limit our attention to formulae uniform in a situation since the AGM and KM are state-based methods, and so there is no need to consider beliefs regarding more than one situation, i.e., situations other than the one currently under consideration.

[12] We can also handle a variant where nothing is believed about the cat sleeping initially by making the first two groups the most plausible.

Definition 8 *(K_t)*
Let t be a ground situation term. We denote a belief state at t by K_t and define it as follows:

$$K_t = \{\phi : \Sigma \models Bel(\phi, t) \text{ and } \phi \text{ is uniform in now}\}$$

It is easily verified that K_t is closed under deduction.

We first define a belief expansion operator ($K_t + \phi$), which returns the set of (uniform) formulae that the agent believes are implied by ϕ at t.

Definition 9 *($K_t + \phi$)*
Let t be a ground situation term and ϕ be a formula uniform in now. We denote the expansion of K_t with ϕ by $K_t + \phi$ and define it as follows:

$$K_t + \phi = \{\psi : \Sigma \models Bel(\phi \supset \psi, t) \text{ and } \psi \text{ uniform in now}\}$$

Next, we define the revision of K_t by A for ϕ ($K_t *_A \phi$) as the belief set held by the agent in the situation that results from performing revision action A for ϕ. In the AGM setting, a revision $K * \phi$ is interpreted as the revision of beliefs K *after learning* ϕ. In our case, we do not know whether ϕ will be true until after performing A. Accordingly, we define a revision of K_t by A for ϕ only in the case that ϕ happens to be true in situation t (i.e., $\phi[t]$ holds).

Definition 10 *($K_t *_A \phi$)*
Let t be a ground situation term, ϕ be a formula uniform in now, and A be a revision action for ϕ. We define the revision of K_t by A for ϕ to be

$$K_t *_A \phi = K_{do(A, t)}$$

whenever $\phi[t]$. If $\neg\phi[t]$, then $K_t *_A \phi$ is undefined.

We now state the relationship with the AGM theory.

Theorem 10 *Let t be a ground situation term, ϕ be a formula uniform in now, and A be a revision action for ϕ. If $K_t *_A \phi$ is defined, then it satisfies AGM postulates (K*1)—(K*4) and (K*6).*

Notice that postulate (K*5) is not satisfied because the agent will end up in inconsistency, if in t there are no B-related situations where ϕ holds. In our framework, the agent is also not capable of recovering from inconsistency. Once everything is believed possible at a situation (i.e., it has no B-related situations), there is no action that can be performed to remedy this. Also note that it does not make sense in our framework to sense (or, for that matter, try to bring about) a formula ϕ known to be necessarily false (i.e., $\models \neg\phi$).

Now, we take the update of a belief state K_t by update action A for formula ϕ to be the set of beliefs held by the agent in the situation that results from performing A. Recall that A causes ϕ to hold.

Definition 11 *Let t be a ground situation term, ϕ be a formula uniform in now, and A be an update action for ϕ. We define the update of K_t by A for ϕ to be:*

$$K_t \diamond_A \phi = K_{do(A, t)}$$

If no action makes ϕ true, then $K_t \diamond_A \phi$ is undefined.

The essential difference between the definitions of revision and update is that the former is effected by sensing (revision) actions while the latter by non-sensing (update) actions and the two are dealt with quite differently in our framework. We now compare with the KM theory.

Theorem 11 *Let t be a ground situation term, ϕ be a formula uniform in now, and A be an update action for ϕ. If $K_t \diamond_A \phi$ is defined, then it satisfies KM postulates (K$\diamond$1)—(K$\diamond$2) and (K$\diamond$4)—(K$\diamond$5).*

Notice that postulate (K$\diamond$3) is not satisfied because an update action for ϕ may have other effects, so despite the fact that the agent believes ϕ beforehand, we cannot guarantee that nothing will change. Boutilier [5] has a problem with this postulate ((U2) in the KM rendering) for similar reasons. In his framework, (update) actions have plausibilities, and the most plausible action explaining the new information is assumed to have taken place. It could be that this action has other effects. To satisfy this postulate, he introduces a *null event* and considers a model in which this is the most plausible event at any world.

In our framework, iterated revision corresponds to the performing of at least two consecutive revision actions. We now show that there is some correspondence with the Darwiche and Pearl account of iterated belief revision.

Theorem 12 *Let t be a ground situation term, ϕ and ψ be formulae uniform in now, A be a revision action for ϕ, and B be a revision action for ψ. Then if $*_A$ and $*_B$ are defined, they satisfy postulates (DP1), (DP3) and (DP4).*[13]

Interestingly, changes of the type described by (DP2) are not defined according to our view of belief revision. In the case where sensing ψ allows us to conclude $\neg\phi$, it is not defined to first sense for ψ and subsequently to sense for ϕ.

6.3 Previous Work

Belief change in the situation calculus has already been dealt with by Scherl and Levesque [19]. However, as noted previously, while they can handle belief update, they are limited to belief expansion. del Val and Shoham [7] also address the issue of belief change in the situation calculus, and their theory deals with both revision and update. However, they cannot represent nested belief and consequently cannot deal with the issues of belief introspection and mistaken belief.

[13] Applying Definition 10, we have that $(K *_A) *_B \psi = K_{do(B,do(A,t))}$ and $K *_B \psi = K_{do(B,t)}$.

There are a variety of frameworks that accommodate both belief revision and belief update. As noted, this is one strength of the proposal by del Val and Shoham [7]. In a more traditional belief change setting, Boutilier [3] also provides a general framework that allows for both these forms of change. However, this framework cannot deal with introspection in the object language. One approach that supports both belief revision and update and also handles introspection is Friedman and Halpern [8]. Their approach to revision and update is fairly standard, but set within a very general modal logic framework that combines operators for knowledge, belief (interpreted a using plausibility ordering), and time. But they do not discuss interactions between revision and update and introspection. We also think that it may suffer from some of the problems mentioned in Section 6.1 that prompted us to abandon approaches based on updating plausibilities.

7 Conclusions and Future Work

We have proposed an account of iterated belief change that integrates into a well-developed theory of action in the situation calculus [18]. This has some advantages, in that previous work on the underlying theory can be exploited for dealing with issues such as solving the frame problem, performing automated reasoning about the effects of actions, specifying and reasoning about complex actions, etc. Our framework supports the introspection of beliefs and ensures that the agent is aware of when it was mistaken about its beliefs. Our account of iterated belief change differs from previous accounts in that, for us, the plausibility assignment to situations remains fixed over time. The dynamics of belief derives from the dynamics of the B modality and of the domain-dependent fluents. We showed that our theory satisfies the majority of the AGM, KM, and DP postulates.

Our approach does have some limitations. In this paper, we have only looked at cases of belief change where the sensors are accurate, so that the agent only revises its beliefs by sentences that are actually true. It is the case that our successor state axiom for B ensures that the agent believes the output of its sensor after sensing. Also, our guarded sensed fluent axioms allow only hard (but context-dependent) constraints to be specified between the output of the sensor and the associated fluent; one cannot state that the sensor is only correct with a certain probability. However, we can also use beliefs to correlate sensor values to the associated fluents instead of guarded sensed fluent axioms. Thus, we could specify that the agent prefers histories where the sensors agree with the associated fluents more often to histories where they agree less often. We will explore this approach in future work. Note that Bacchus et al. [2] have a probabilistic account of noisy sensors in the situation calculus.

In Theorem 10, we saw that our framework captures some, but not all, of the AGM revision postulates. In particular, the agent may end up believing everything after a revision by a consistent formula ϕ, if none of its B-alternatives satisfies ϕ, violating (K*5). This, together with the fact that we never update the plausibility assignment, may suggest that our account has limited expressiveness. But we maintain that this is not the case. The example of Section 6.1 shows that we can handle some cases where a plausibility assignment update seems to be required. As well, we can construct theories where the (K*5) postulate does hold. This is done by ensuring that the B relation contains enough situations initially.[14] The need to ensure that enough epistemic alternatives are initially present if one wants to avoid inconsistency is not specific to our approach. In most frameworks, a similar issue arises with respect to revision by conjunctive observations. In future work, we will investigate the expressive limits of our framework.

We could also extend the framework by having multiple agents that act independently and impart information to each other. Instead of beliefs changing only through sensing, they would also change as a result of *inform* actions. Shapiro et al. [20] provide a framework for belief expansion resulting from the occurrence of inform actions in the situation calculus, which we would like to generalize to handle belief revision.

Lakemeyer and Levesque [12] incorporate the logic of *only knowing* into the Scherl and Levesque framework of belief update and expansion. The traditional belief (and knowledge) operator specifies formulae that are believed (or known) by the agent, but there could be others. The 'only knows' operator is used to describe *all* that the agent knows, i.e., a formula that corresponds exactly to the knowledge state of the agent. In future work, we would like to define an analogous 'only believes' operator that could be used to describe exactly what the agent believes in a framework that supports belief revision as well as belief expansion.

Acknowledgments

The authors would like to gratefully acknowledge Craig Boutilier and Jim Delgrande for valuable comments on an earlier version of this paper. Financial support for the Canadian authors was gratefully received from the Natural Sciences and Engineering Research Council of Canada, and the Institute for Robotics and Intelligent Systems of Canada. The Australian author is partially supported by a Macquarie University MUNS grant.

[14] Note also that when no guarded sense fluent axiom is applicable, the value of an *SF* fluent is unconstrained and can vary freely. There can be B-alternatives that differ only in the value of the *SF* fluent after some sequence of sensing actions has been performed. By modifying the definition of $*_A$, and ensuring that there are enough B alternatives, we *can* model iterated revisions involving contradictory information where the agent's beliefs remain consistent, i.e., where $K * \phi * \neg\phi \not\models \bot$.

References

[1] Carlos E. Alchourrón, Peter Gärdenfors and David Makinson. On the the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50: 510–533, 1985.

[2] Fahiem Bacchus, Joseph Y. Halpern, and Hector J. Levesque. Reasoning about noisy sensors and effectors in the situation calculus. *Artificial Intelligence*, 111(1–2):171–208, 1999.

[3] Craig Boutilier. Generalized update: Belief change in dynamic settings. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, 1995.

[4] Craig Boutilier. Iterated revision and minimal change of conditional beliefs. *Journal of Philosophical Logic*, 25(3):262–305, 1996.

[5] Craig Boutilier. Abduction to plausible causes: An event-based model of belief update. *Artificial Intelligence*, 83(1):143–166, 1996.

[6] Adnan Darwiche and Judea Pearl. On the logic of iterated belief revision. *Artificial Intelligence*, 89(1–2):1–29, 1997.

[7] Alvaro del Val and Yoav Shoham. A unified view of belief revision and update. *Journal of Logic and Computation*, 4:797–810, 1994.

[8] Nir Friedman and Joseph Y. Halpern. A knowledge-based framework for belief change, part II: Revision and update. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR-94)*, pages 190–201, 1994.

[9] P. Gärdenfors. *Knowledge in Flux*. The MIT Press, Cambridge, MA, 1988.

[10] Giuseppe De Giacomo and Hector J. Levesque. Progression using regression and sensors. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 160–165, 1999.

[11] Hirofumi Katsuno and Alberto O. Mendelzon. On the difference between updating a knowledge base and revising it. In Peter Gärdenfors, editor, *Belief Revision*, pages 183–203, Cambridge University Press, 1992.

[12] Gerhard Lakemeyer and Hector J. Levesque. $\mathcal{AOL}$: a logic of acting, sensing, knowing, and only knowing. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR-98)*, pages 316–327, 1998.

[13] Hector J. Levesque. What is planning in the presence of sensing? In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1139–1146, August 1996.

[14] David Lewis. *Counterfactuals*. Harvard University Press, Cambridge, Massachusetts, 1973.

[15] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of Artificial Intelligence. In Bernard Meltzer and Donald Michie, editors, *Machine Intelligence 4*. Edinburgh University Press, 1969.

[16] Robert C. Moore. A formal theory of knowledge and action. In Jerry R. Hobbs and Robert C. Moore, editors, *Formal Theories of the Common Sense World*, pages 319–358. Ablex Publishing, Norwood, NJ, 1985.

[17] Pavlos Peppas, Abhaya C. Nayak, Maurice Pagnucco, Norman Y. Foo, Rex Kwok, and Mikhail Prokopenko. Revision vs. update: Taking a closer look. In W. Wahlster, editor, *Proceedings of the Twelfth European Conference on Artificial Intelligence (ECAI-96)*, pages 95–99, August 1996.

[18] Raymond Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, San Diego, CA, 1991.

[19] Richard B. Scherl and Hector J. Levesque. The frame problem and knowledge-producing actions. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 689–695, July 1993.

[20] Steven Shapiro, Yves Lespérance, and Hector J. Levesque. Specifying communicative multi-agent systems. In Wayne Wobcke, Maurice Pagnucco, and Chengqi Zhang, editors, *Agents and Multi-Agent Systems — Formalisms, Methodologies, and Applications*, volume 1441 of *LNAI*, pages 1–14. Springer-Verlag, Berlin, 1998.

[21] Wolfgang Spohn. Ordinal conditional functions: A dynamic theory of epistemic states. In W. Harper and B. Skyrms, editors, *Causation in Decision, Belief Change, and Statistics*, pages 105–134. Kluwer Academic Publishers, Dordrecht, 1988.

[22] Mary-Anne Williams. Transmutations of knowledge systems. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR-94)*, pages 619-629, 1994.

Ontology-Based Semantics

Mihai Ciocoiu
Dept. of Computer Science
and Institute for Systems Research
University of Maryland
College Park, MD 20742
mihaic@cs.umd.edu

Dana S. Nau
Dept. of Computer Science
and Institute for Systems Research
University of Maryland
College Park, MD 20742
nau@cs.umd.edu

Abstract

We consider the problem of providing semantics for declarative languages, in a way that would be useful for enabling automated knowledge exchange. If we only have one (first order) language, we can formalize what we mean for a set of sentences to be a translation of another, by requiring that the two sets share the same models. However, in order to formalize translation for the case where the two sets of sentences are of different languages, we need a different notion of semantics, capable of overcoming the language barrier. We introduce *Ontology-Based Semantics* with this purpose in mind. We show how ontologies can be used to make implicit assumptions explicit, and how they are integrated in our semantics in order to restrict the set of models a set of sentences has. We show how Ontology-Based Models can be used to formally define knowledge translation for the different language case in a similar way ordinary models can be used to define translation for the one language situation. We also provide a syntactical characterization of knowledge translation, that can be used as an effective procedure to check translatability, and we prove it to be sound and complete with respect to our semantic definition of translation.

1 INTRODUCTION

The ability to translate knowledge from/to different representation languages is an important ingredient for building powerful AI systems, by easing the difficult and time-consuming task of knowledge base construction, and facilitating knowledge sharing among existing ones. In this paper we consider the problem of using formal ontologies for providing semantics to declarative languages, in a way that would be useful for enabling automated knowledge exchange.

Using formal ontologies has been proposed [Gruber 1991] as a solution for managing the inherent heterogeneity present in knowledge from different sources. Different approaches vary in their definition of what a formal ontology is, ranging from taxonomic hierarchies of classes [Campbell et al.], to vocabularies of terms defined by human-readable text, together with sets of formal constraining axioms [Gruber 1993]. Another distinction is the level of commitment of the communicating agents with respect to the shared ontology, varying from having all agents commit to a single common ontology — the standardization approach — to having a network of mediators and facilitators that enable translation among agents' different ontologies [Shave 1997, Gray et al. 1997].

For our purpose we will adopt the logical theory view of an ontology, and the constraining axioms will play a crucial role in defining our semantics. While we allow the communicating agents to have their own declarative languages and ontologies, we will require the existence of a common ontology expressive enough to interpret the concepts in all agents' ontologies. We will also require for a declarative language L that a function σ can be specified that converts sentences of L to sentences of a first order language $\mathcal{L}$. (of course this limits the method's applicability to languages that are not (strictly) more expressive than FOL). Here are some examples of questions we want to address:

- Consider a declarative language L that has a construct like `(mother Bill Anne)`. What would models of this construct look like?
- Consider now the problem of translating sets of sentences among two declarative languages L_1

and L_2. What exactly do we mean when we say that a set S_2 of L_2 sentences is a translation of a set S_1 of L_1 sentences?

For the simplest case, in which we have just one first-order language L and we are considering two sets of L-sentences S_1 and S_2, the obvious solution is to say that S_2 is a *translation* of S_1 if it has the same set of consequences (i.e., $Cn(S_2) = Cn(S_1)$), or equivalently, if S_1 and S_2 share the same set of models (i.e., $\mathfrak{A} \models S_1 \Leftrightarrow \mathfrak{A} \models S_2$ for all $\mathfrak{A}$). One could extend this further by saying that S_2 is a *partial translation* of S_1 if the consequences of S_2 are also consequences of S_1 (i.e., $Cn(S_2) \subseteq Cn(S_1)$), or equivalently, if all models of S_1 are models of S_2 (i.e., $\mathfrak{A} \models S_1 \Rightarrow \mathfrak{A} \models S_2$ for all $\mathfrak{A}$).

Unfortunately, a direct extension of this idea for sets of sentences of two *different* first order languages L_1 and L_2 will not work the way we would like. One problem is that for intuitively similar concepts (and thus ones that we would like translatable) their representations in the two languages might use combinations of functions/predicates of different arities, such as functions in one language and predicates in the other (or any combinations thereof). Thus, models of sets of sentences in the different languages will be different, even if the sets of sentences are intuitively equivalent. For two different arbitrary declarative languages, defining translation is even harder, since we don't even have the notion of a model.

What this paper tries to do is to specify a way for defining models of sets of sentences of arbitrary declarative languages, so that these models can be used to define translation in the same fashion as for the one language situation above.

2 IMPLICIT ASSUMPTIONS

Consider a declarative language L that has a construct like `(mother Bill Anne)`. What would we want models of this construct to look like?

One option would be to follow a database approach: use a closed world assumption, proclaim that we are speaking of a universe with only two persons (`Anne` and `Bill`), and that the motherhood relation holds only for $\langle$`Bill, Anne`$\rangle$. In this case, we would have a single model $\mathfrak{A} = (\{$`Bill, Anne`$\}, \{\langle$`Bill, Anne`$\rangle\})$. However, this approach would not allow us to define partial translation the way we would like, and would prohibit translating bits and pieces of information from more expressive languages into less expressive ones. Instead, we would prefer the semantics of `(mother Bill Anne)` to be "the universe includes `Anne` and `Bill` and maybe other persons and the motherhood relation holds for $\langle$`Bill, Anne`$\rangle$ and maybe for some other pairs," like the semantics for FOL is usually defined.

The first temptation would be to define the relation σ so that:

1. σ maps an L construct such as `(mother ?x ?y)` into a predicate such as Mother(x, y) of the first order language $\mathcal{L}$;
2. the models of a set of L sentences (e.g., $S_1 =$ {`(mother Bill Anne)`}) are in fact the models of the set $\Sigma_1 = \{\sigma(s) : s \in S_1\}$; (e.g., the models of {Mother(Bill, Anne)}).

This would allow other wanted structures, that include different persons, to be considered, like $\mathfrak{B} =$ ({`Bill, Anne, Cathy`}, {$\langle$`Bill, Anne`$\rangle$, $\langle$`Anne, Cathy`$\rangle$}). However, it would also allow unwanted models to creep in, such as:

$\mathfrak{C} =$ ({`Bill, Anne`}, {$\langle$`Bill, Anne`$\rangle$, $\langle$`Anne, Anne`$\rangle$}).

$\mathfrak{D} =$ ({`Bill, Anne, Cathy`}, {$\langle$`Bill, Anne`$\rangle$, $\langle$`Bill, Cathy`$\rangle$}).

$\mathfrak{E} =$ ({`Bill, Anne, Cathy`}, {$\langle$`Bill, Anne`$\rangle$, $\langle$`Anne, Cathy`$\rangle$, $\langle$`Cathy, Bill`$\rangle$}).

The problem is that there are implicit assumptions in the language L, such as the fact that a person cannot be her own mother, that one cannot have two different mothers, etc. These assumptions need to be made explicit in order to define the correct set of models for `(mother Bill Anne)`.

Making such assumptions explicit has become known in AI as building the domain ontology [Gruber 1991], and it holds the promise of enabling knowledge exchange.

The above considerations lead us to explicitly constructing a motherhood ontology Ω into $\mathcal{L}$ and taking the models of the L sentence `(mother Bill Anne)` to be the models of the $\mathcal{L}$ theory having as axioms the sentence's image through σ together with the ontology itself, i.e. the models of $\Omega \cup$ {Mother(Bill, Anne)}. Will this do?

Well, it will give us the models we wanted, but will not help much with translation. If some other language defines in its ontology a motherhood relation, then conceivably an automated translation procedure would identify the motherhood predicates as mutually translatable, but it would also translate to mother-

hood all predicates that satisfy the motherhood ontology. For example, the successor predicate that holds between an integer number and its successor satisfies all of the motherhood axioms, and so an automated translation procedure will consider a (partial) translation of `(s ?x ?y)` to `(mother ?x ?y)`. (The translation will only be one-way, since the successor relation satisfies additional constraints, being in fact a bijective function, as opposed to the motherhood, which is only surjective.)

A much better idea is to *share* the motherhood ontology. Sharing ontologies will greatly simplify the task of a semantic-based automated translation procedure and also have the additional benefit of simplifying the process of writing ontologies, by enabling the reuse of already-existing components. In the next section we will present a way to define semantics that makes use of the ontology sharing idea.

3 ONTOLOGY-BASED SEMANTICS

3.1 LOGICAL RENDER

As we mentioned earlier, we are interested in providing semantics for *declarative languages*, which for our purpose are languages L such that a function σ can be specified that converts sentences of L to sentences of a first order language $\mathcal{L}$.

We call such a function σ a *logical rendering* function, and the image Σ of a set S of L sentences through σ the *logical render* of S through σ.

Coming back to our motherhood example, the *logical rendering* function σ will convert instances of `(mother ?x ?y)` to corresponding instances of the $\mathcal{L}$ predicate Mother(x, y).

3.2 INTERPRETATIONS

To simplify our definitions we will restrict ourselves to first order languages $\mathcal{L}$ that contain no function symbols. Note that this is not a reduction of expressivity, since any formula of a first order language $\mathcal{L}$ that includes function symbols can be converted to a formula of a language $\mathcal{L}'$ similar to $\mathcal{L}$ but which has no function symbols and has additional $(n+1)$-ary predicates corresponding to each n-ary function of $\mathcal{L}$.

Our notion of interpretation is the restriction to predicate calculus of the standard mathematical one, as it appears in [Enderton 1972].

Definition An *interpretation* π of a function-free language $\mathcal{L}$ into a theory T of language $\mathcal{L}_\Omega$ is a function on the set of parameters of $\mathcal{L}$ such that:

1. π assigns to $\forall$ a formula $\pi_\forall$ of $\mathcal{L}_\Omega$ in which at most one variable v_1 occurs free, such that $T \models \exists v_1 \pi_\forall$.
2. π assigns to each n-place predicate symbol P a formula π_P of $\mathcal{L}_\Omega$ in which at most n variables $v_1, \ldots, v_n$ occur free.

Definition An *interpretation* φ^π of a $\mathcal{L}$-formula φ is recursively defined in the obvious way: i.e. if φ is an atomic formula P, its interpretation is the formula π_P applied to the same set of constants/variables; otherwise $(\neg\varphi)^\pi$ is $(\neg\varphi^\pi)$, $(\varphi \rightarrow \psi)^\pi$ is $(\varphi^\pi \rightarrow \psi^\pi)$, $(\forall x\varphi)^\pi$ is $\forall x(\pi_\forall(x) \rightarrow \varphi^\pi)$, etc.

Definition An *interpretation* π of a theory T_0 of language $\mathcal{L}$ into a theory T of language $\mathcal{L}_\Omega$ is an interpretation π of the language $\mathcal{L}$ into T such that for all $\mathcal{L}$-sentences φ, $\varphi \in T_0 \Rightarrow \varphi^\pi \in T$.

3.3 EXPLANATIONS

Definition Given a domain ontology Ω expressed as a set of $\mathcal{L}_\Omega$ sentences, a theory T of $\mathcal{L}_\Omega$ is called a *domain theory* for Ω iff $T \models \Omega$.

Definition Given an interpretation π, a logical rendering function σ and an ontology Ω (expressed as a set of $\mathcal{L}_\Omega$ sentences), an $\mathcal{L}_\Omega$ *explanation* of a set S of L sentences is a $\mathcal{L}_\Omega$ domain theory T for Ω, such that if Σ is the logical render of S through σ, $T \models \Sigma^\pi$.

Intuitively, an $\mathcal{L}_\Omega$ *explanation* of a set S of sentences of language L is a theory of $\mathcal{L}_\Omega$ that has among its axioms the interpretation of the rendering of S, with the concepts that appear in them "explained" by ontology Ω. Going back to our example, an explanation of `(mother Bill Anne)` is a theory that has Mother(Bill, Anne) and the motherhood ontology as axioms.

3.4 ONTOLOGY-BASED MODELS

Given a model $\mathfrak{A}$ of an explanation T of a set of L sentences S, we can extract from it a model ${}^\pi\mathfrak{A}$ of the render Σ of S, that has the desired property of obeying the additional constraints imposed by the ontology Ω. Namely, let

1. $|{}^\pi\mathfrak{A}|$ = the set defined in $\mathfrak{A}$ by $\pi_\forall$;
2. $P^{{}^\pi\mathfrak{A}}$ = the relation defined in $\mathfrak{A}$ by π_P, restricted to $|{}^\pi\mathfrak{A}|$.

${}^\pi\mathfrak{A}$ is called an Ontology-Based Model of S (written ${}^\pi\mathfrak{A} \models^\Omega_{\sigma,\pi} S$).

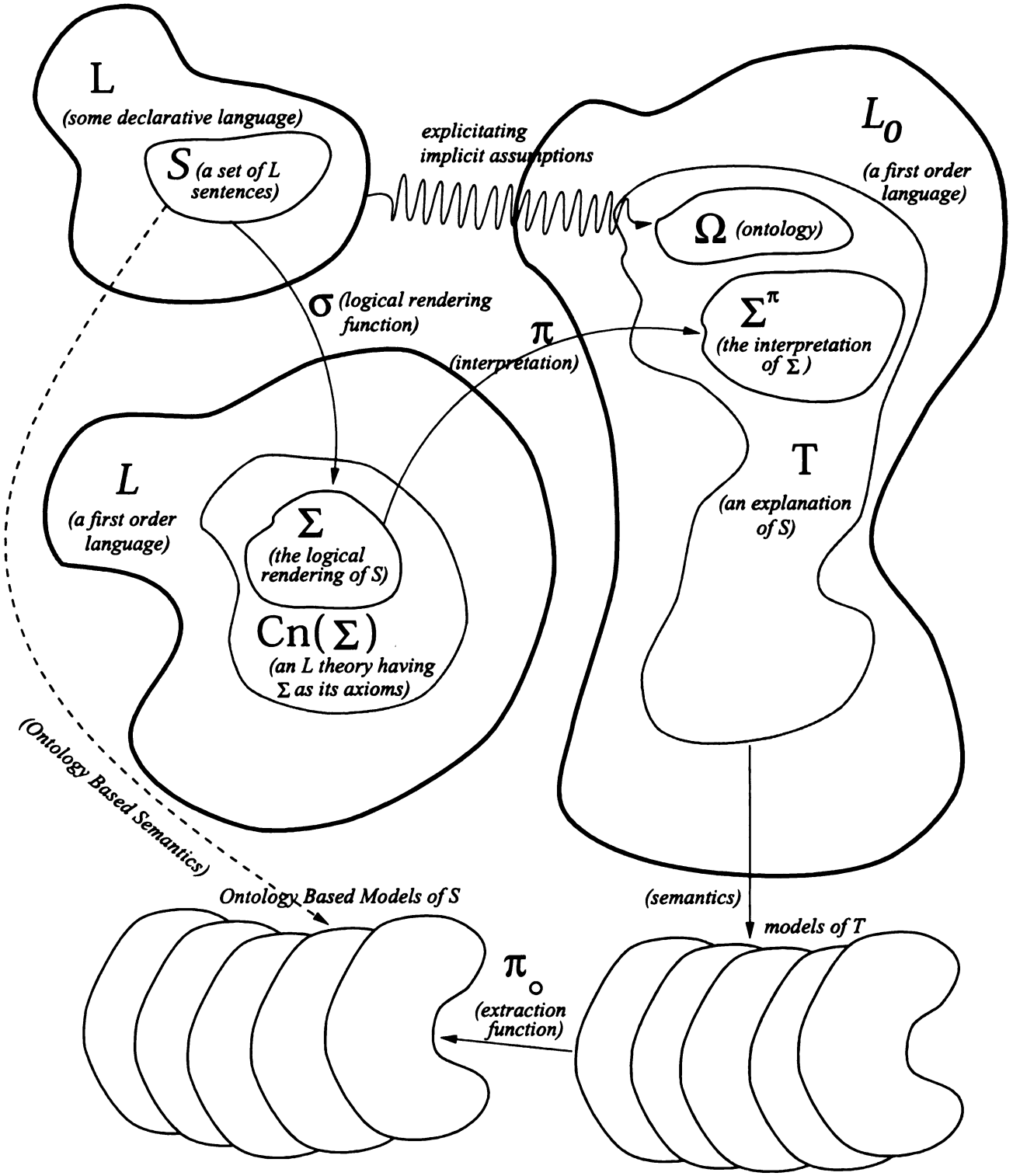


Figure 1: Ontology-Based Models

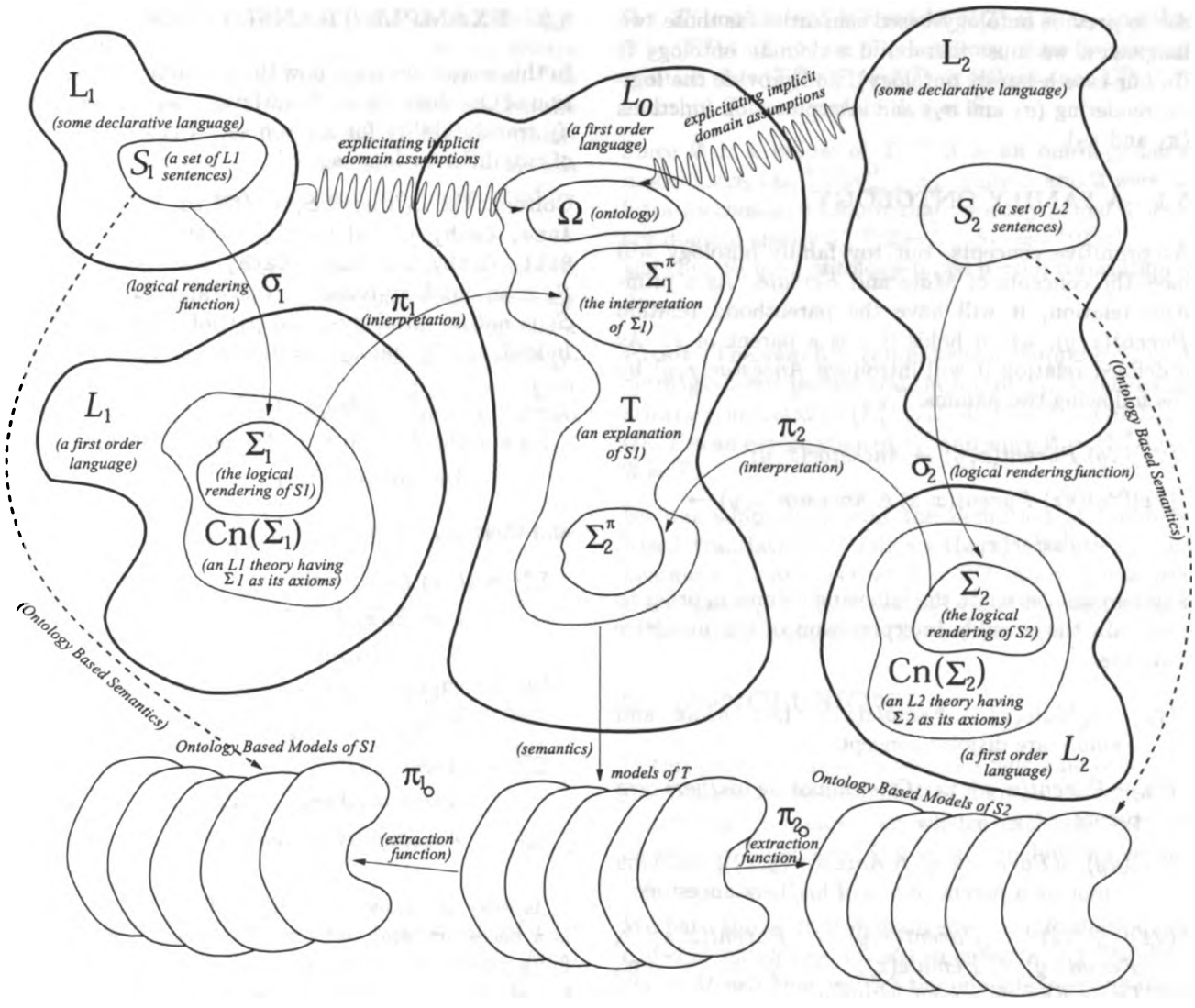


Figure 2: Ontology-Based Translation

4 ONTOLOGY-BASED TRANSLATION

Ontology-Based Models allow a definition of translation for the different language situation in the same way that ordinary models allowed it for the single-language case.

Suppose we are given two declarative languages L_1 and L_2, a domain ontology Ω (expressed as a set of sentences in language $\mathcal{L}_\Omega$), rendering functions σ_1 and σ_2, and interpretations π_1 and π_2. Then a set S_2 of L_2 sentences is an *Ontology-Based Partial Translation* of a set S_1 of L_1 sentences iff for every model $\mathfrak{A}$ of every explanation T of S_1, ${}^{\pi_1}\mathfrak{A} \models^{\Omega}_{\sigma_1,\pi_1} S_1 \Rightarrow {}^{\pi_2}\mathfrak{A} \models^{\Omega}_{\sigma_2,\pi_2} S_2$.

Similarly, suppose we are given two declarative languages L_1 and L_2, a domain ontology Ω (expressed as a set of sentences in language $\mathcal{L}_\Omega$), rendering functions σ_1 and σ_2, and interpretations π_1 and π_2. Then a set S_2 of L_2 sentences is an *Ontology-Based Translation* of a set S_1 of L_1 sentences iff S_2 is an Ontology Based Partial Translation of S_1 and S_1 is an Ontology-Based Partial Translation of S_2.

5 AN EXAMPLE TRANSLATION

Suppose we have a declarative language L_1 that has a construct like `(GM ?x ?y)` whose intended semantics is "`?y` is an grandmother of `?x`", and another declarative language L_2 that has a construct `Y anc X` whose intended semantics is "`Y` is an ancestor of `X`". In or-

der to provide ontology-based semantics for those two languages, we must first build a domain ontology Ω (in our case a family ontology), and provide the logical rendering (σ_1 and σ_2) and interpretation functions (π_1 and π_2).

5.1 A FAMILY ONTOLOGY

As primitive concepts, our toy family ontology will have the concepts of *Male* and *Female*. As a primitive relation, it will have the parenthood relation $Parent(x, y)$, which holds if y is a parent of x. As a defined relation it will introduce $Ancestor(x, y)$ by the following two axioms:

$$(\forall x)(\forall y)\ Parent(x, y) \rightarrow Ancestor(x, y)$$

$$(\forall x)(\forall y)(\forall z)\ Parent(x, z) \wedge Ancestor(z, y) \rightarrow Ancestor(x, y)$$

Suppose we also write the following axioms in order to constrain the possible interpretation of the primitive concepts:

- $(\forall x)\ \neg(Male(x) \wedge Female(x))$; i.e., *Male* and *Female* are disjoint concepts.
- $(\forall x)\ \neg Parent(x, x)$; i.e., One cannot be his/hers own parent.
- $(\forall x)(\forall y)\ \neg(Parent(x, y) \wedge Ancestor(y, x))$; i.e., One cannot be a parent of one of his/hers ancestors.
- $(\forall x)(\forall y)(\forall z)\ (Parent(x, y) \wedge Parent(x, z) \wedge Female(y) \wedge Female(z)) \rightarrow y = z$; i.e., One's *Female* parent is unique.
- $(\forall x)(\forall y)(\forall z)\ (Parent(x, y) \wedge Parent(x, z) \wedge Male(y) \wedge Male(z)) \rightarrow y = z$; i.e., One's *Male* parent is unique.

5.2 LOGICAL RENDERING AND INTERPRETATION FUNCTIONS

The logical rendering functions for this simple example would just convert the given constructs into the corresponding first-order atomic sentences, i.e. the logical render of `(GM ?x ?y)` would be the $\mathcal{L}_1$ atomic sentence $GM(x, y)$ and the logical render of `Y anc X` would be the $\mathcal{L}_2$ atomic sentence $Anc(x, y)$.

The interpretation of $GM(x, y)$ will be a $\mathcal{L}_\Omega$ formula having at most two free variables, in our case $(\exists z)\ Parent(x, z) \wedge Parent(z, y) \wedge Female(y)$.

Finally, the interpretation of $Anc(x, y)$ must be a $\mathcal{L}_\Omega$ formula having at most two free variables, in our case $Ancestor(x, y)$.

5.3 EXAMPLE TRANSLATION

In this section we show how the syntactic characterization of Ontology-Based Translation can be used to verify translatability for a given set of ground sentences of two different languages.

Consider the L_1 theory $S_1 = \{$`(GM Bill, Anne), (GM Anne, Cathy)`$\}$, and the L_2 theory $S_2 = \{$`Anne anc Bill, Cathy anc Anne, Cathy anc Bill`$\}$. Then S_2 is an ontology-based partial translation of S_1, but S_1 is *not* an ontology-based partial translation of S_2. Indeed, the $\mathcal{L}_1$ and $\mathcal{L}_2$ logical renders of S_1 and S_2 are:

$$\Sigma_1 = \{GM(Bill, Anne),\ GM(Anne, Cathy)\} \text{ and}$$
$$\Sigma_2 = \{Anc(Bill, Anne),\ Anc(Anne, Cathy),\ Anc(Bill, Cathy)\}$$

and their $\mathcal{L}_\Omega$ interpretations are:

$$\Sigma_1^{\pi_1} = \{(\exists z)\ Parent(Bill, z) \wedge Parent(z, Anne) \wedge Female(Anne),\ (\exists z)\ Parent(Anne, z) \wedge Parent(z, Cathy) \wedge Female(Cathy)\} \text{ and}$$
$$\Sigma_2^{\pi_2} = \{Ancestor(Bill, Anne),\ Ancestor(Anne, Cathy),\ Ancestor(Bill, Cathy)\}$$

It is easy to show that $(\Sigma_1^{\pi_1} \cup \Omega) \vdash \Sigma_2^{\pi_2}$, and this is a necessary and sufficient condition (see Section 6 for a proof) for S_2 to be an ontology-based partial translation of S_1. However, the reverse is not true. S_1 is not an ontology-based partial translation of S_2. To see this, consider an explanation T of S_2,

$$T = Cn(\Omega \cup \{Parent(Bill, Anne),\ Parent(Anne, Cathy)\})$$

and a model of it:

$$\mathfrak{A} = (\{Anne, Bill, Cathy\},$$
$$Parent^{\mathfrak{A}} = \{\langle Bill, Anne\rangle, \langle Anne, Cathy\rangle\},$$
$$Male^{\mathfrak{A}} = \{Bill\},$$
$$Female^{\mathfrak{A}} = \{Anne, Cathy\},$$
$$Ancestor^{\mathfrak{A}} = \{\langle Bill, Anne\rangle, \langle Anne, Cathy\rangle, \langle Bill, Cathy\rangle\}).$$

Note that ${}^{\pi_2}\mathfrak{A} \models^{\Omega}_{\sigma_2, \pi_2} S_2$, but ${}^{\pi_1}\mathfrak{A} \not\models^{\Omega}_{\sigma_1, \pi_1} S_1$, so S_1 is not an ontology-based partial translation of S_2. Also note that neither $S_1' =$ `{(GM Bill Cathy)}` nor any other

non-void set S_1 of L_1 sentences could be an ontology-based partial translation of S_2, since we can always construct models $\mathfrak{A}$ of S_2 explanations such that ${}^{\pi_1}\mathfrak{A}$ are not ontology-based models of S_1, by violating the sex/parenthood constraints necessary for the grandmotherhood relation of L_1. (Such a class of models would be the ones which interpret the motherhood relation by the void set.)

In practice, an automated inference procedure could be used on the domain ontology and the logical rendering and interpretation functions, in order to precompile a generic rule of the form "?y anc ?x is an ontology-based translation of (GM ?x ?y)", and then use it for generating efficient direct translators among the two languages.

6 A SYNTACTIC CHARACTERIZATION

Theorem 6.1 *(Soundness and Completeness of Syntactic Characterization)*

A set S_2 of L_2-sentences is an ontology-based partial translation *of a set S_1 of L_1-sentences (with respect to rendering functions σ_1 and σ_2 and interpretations π_1 and π_2) iff*

$$(\Sigma_1^{\pi_1} \cup \Omega) \vdash \Sigma_2^{\pi_2}$$

where Σ_1 and Σ_2 are the logical renders of S_1 and S_2 through σ_1 and σ_2.

Lemma 6.2 *If ${}^{\pi}\mathfrak{A}$ is an ontology-based model of a set of sentences S with respect to logical rendering function σ and interpretation π, ${}^{\pi}\mathfrak{A} \models^{\Omega}_{\sigma,\pi} S$, then $\mathfrak{A}$ is a model of the interpretation Σ^{π} of the logical render Σ of S, i.e. $\mathfrak{A} \models \Sigma^{\pi}$.*

Proof (Lemma 6.2) Suppose $\mathfrak{A} \not\models \Sigma^{\pi}$. By definition of an ontology-based model, there must exist an $\mathcal{L}_\Omega$ explanation T of S, such that $T \models \Omega$ and $T \models \Sigma^{\pi}$, and a model $\mathfrak{B}$ of T such that ${}^{\pi}\mathfrak{B} = {}^{\pi}\mathfrak{A}$. (If such a model doesn't exist, then ${}^{\pi}\mathfrak{A}$ cannot be an ontology-based model of S.)

Since T is a theory and $T \models \Sigma^{\pi}$, it must be the case that $\Sigma^{\pi} \subseteq T$. Since $\mathfrak{B} \models T$, $\mathfrak{B}$ is a model for Σ^{π}, $\mathfrak{B} \models \Sigma^{\pi}$. Since $\mathfrak{A} \not\models \Sigma^{\pi}$, there must exist a sentence $\gamma^{\pi} \in \Sigma^{\pi}$ such that $\mathfrak{A} \not\models \gamma^{\pi}$. It can be proved by induction on the structure of γ that this cannot be the case.

Proof (Theorem 6.1, soundness) Suppose $(\Sigma_1^{\pi_1} \cup \Omega) \vdash \Sigma_2^{\pi_2}$. Consider an arbitrary domain theory T, and an arbitrary model $\mathfrak{A}$ of T. If ${}^{\pi_1}\mathfrak{A} \models^{\Omega}_{\sigma_1,\pi_1} S_1$, then by Lemma 6.2, $\mathfrak{A} \models \Sigma_1^{\pi_1}$. Since T is a domain theory, $\Omega \subseteq T$; and since $\mathfrak{A}$ is a model of T, $\mathfrak{A} \models \Omega$; and thus $\mathfrak{A} \models (\Sigma_1^{\pi_1} \cup \Omega)$. Since T is a theory, then by our supposition that $(\Sigma_1^{\pi_1} \cup \Omega) \vdash \Sigma_2^{\pi_2}$, it follows that $\Sigma_2^{\pi_2} \subseteq T$, and thus T is an explanation of S_2.

Since $\mathfrak{A}$ is a model of T, ${}^{\pi_2}\mathfrak{A}$ is an ontology-based model for S_2 i.e., ${}^{\pi_2}\mathfrak{A} \models^{\Omega}_{\sigma_2,\pi_2} S_2$. Since T and $\mathfrak{A}$ were arbitrarily chosen, it follows that for every model $\mathfrak{A}$ of every domain theory T, ${}^{\pi_1}\mathfrak{A} \models^{\Omega}_{\sigma_1,\pi_1} S_1 \Rightarrow {}^{\pi_2}\mathfrak{A} \models^{\Omega}_{\sigma_2,\pi_2} S_2$, and thus S_2 is an ontology-based partial translation of S_1.

Proof (Theorem 6.1, completeness) Suppose S_2 is an ontology-based partial translation of S_1. Consider an arbitrary model $\mathfrak{A} \models (\Sigma_1^{\pi_1} \cup \Omega)$ and let $T = Cn(\Sigma_1^{\pi_1} \cup \Omega)$. T is an explanation of S_1; and since $\mathfrak{A} \models (\Sigma_1^{\pi_1} \cup \Omega)$, $\mathfrak{A} \models T$.

By our supposition and the definition of ontology-based translation, it follows that ${}^{\pi_2}\mathfrak{A} \models^{\Omega}_{\sigma_2,\pi_2} S_2$. By Lemma 6.2 it follows that $\mathfrak{A} \models \Sigma_2^{\pi_2}$. Since $\mathfrak{A}$ was arbitrary chosen, $(\Sigma_1^{\pi_1} \cup \Omega) \models \Sigma_2^{\pi_2})$; and by completeness of first order deduction, $(\Sigma_1^{\pi_1} \cup \Omega) \vdash \Sigma_2^{\pi_2}$.

7 CONCLUSIONS

In this paper we have presented a new kind of semantics, called *Ontology-Based Semantics*, intended for facilitating automated knowledge exchange between declarative languages. Our results include the following:

We have shown how domain specific information, encoded as ontologies, is used in constructing *Ontology-Based Models* that restrict the possible interpretations a set of sentences can have.

We have shown how Ontology-Based Models can be used to formally define knowledge translation for the different-language case, in a way similar to how ordinary models can be used to define translation for the one-language situation.

We have provided a syntactical characterization of knowledge translation, that can be used as an effective procedure to check translatability, and we have proved it to be sound and complete with respect to our semantic definition of translation.

The principal benefit of our semantics is that it provides a formal foundation for reasoning about the properties of systems that do automated knowledge translation based on ontology sharing. As part of our collaboration with NIST on their Process Specification Language (PSL) project [Schlenoff et al. 1999, Schlenoff et al. 1999], we plan to develop a system that would be able to automatically generate efficient

translators based on declarative languages' ontology-based semantics, specified as a logical rendering function and a PSL interpretation.

Acknowledgements

This work was supported in part by the following grants and contracts: National Institute for Standards and Technology 70NANB6H0147, Army Research Laboratory DAAL01-97-K0135, Air Force Research Laboratory F306029910013, and National Science Foundation DMI-9713718.

References

[Campbell et al.] Alistair E. Campbell, Hans Chalupsky, and Stuart C. Shapiro. Ontological Mediation: An Analysis. Unpublished manuscript.

[Enderton 1972] H. B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, 1972.

[Gray et al. 1997] P. M. D. Gray et al. KRAFT: Knowledge Fusion from Distributed Databases and Knowledge Bases. In *The 8th International Conference and Workshop on Database and Expert Systems Applications*, Toulouse, France, September 1997.

[Gruber 1991] Thomas R. Gruber. The Role of Common ontology in achieving sharable, reusable knowledge bases. In Richard Fikes, James A. Allen, and Erik Sandewall, editors, *Proceedings of the Second International Conference*, Principles of Knowledge Representation and Reasoning, 1991.

[Gruber 1993] Thomas R. Gruber. Toward Principles of the Design of Ontologies Used for Knowledge Sharing. In *International Workshop on Formal Ontology in Conceptual Analysis and Knowledge Representation*, Padova, Italy, 1993.

[Schlenoff et al. 1999] Craig Schlenoff, Mihai Ciocoiu, Don Libes, and Michael Gruninger. Process Specification Language: Results of the First Pilot Implementation In *Proceedings of the 1999 International Mechanical Engineering Congress and Exposition (IMECE)*, November 1999.

[Schlenoff et al. 1999] Craig Schlenoff, Michael Gruninger, and Mihai Ciocoiu. The Essence of Process Specification Submitted, by invitation, to the *Special Issue on Modeling and Simulation of Manufacturing Systems for the Society for Computer Simulation International*, 1999.

[Shave 1997] M. J. R. Shave. Ontological Structures for Knowledge Sharing In *New Review of Information Networking*, 1997.

Supporting Automated Deduction in First-Order Modal Logics

Angelo Montanari and Alberto Policriti
Dipartimento di Matematica e Informatica
Università di Udine
33100 Udine, Italy.

Matteo Slanina
Department of Computer Science
Stanford University
Stanford, CA 94305-9045

Abstract

One of the most important and frequently employed methods to support derivability in modal logics consists in translating them into first-order logic. Such an approach has many advantages, among which its uniformity and the possibility of exploiting well-known deductive engines for first-order logic. Most of the proposed methods can be directly applied to only first-order complete modal logics (i.e. modal logics whose Kripke semantics is first-order axiomatizable). Many efforts have been devoted to deal with more challenging (and widely used) modal logics. As a matter of fact, almost all the translations in the literature work for the propositional case.

In [DMP95], D'Agostino et al. proposed a novel method of translating propositional modal logics into simple set theories. The main feature of such a method is that the use of a set theory (suitable fragment of second-order logic) as target formalism allows a uniform translation of any normal, finitely axiomatizable (poly)modal logic. In this paper, we present an elegant extension of this set-theoretic translation method to an important class of first-order modal logics (by far the most used), namely, objectual logics with rigid designators and fixed domain.

1 INTRODUCTION

Modal logics are at the basis of a variety of knowledge representation and reasoning systems. As an example, a large family of knowledge representation languages, including terminological logics, epistemic logics, and universal modalities, can be represented using graded modalities [HdR95]. Even though several applications confine themselves to propositional modal logics, in many situations there is a quest for a switch to the first-order setting [Fi90].

In this paper, we focus on the fundamental problem of supporting derivability in (first-order) modal logic. One of the most important and frequently employed methods consists in translating modal logic(s) into first-order logic, e.g. [Oh91]. Such an approach has many advantages, among which its uniformity and the possibility of exploiting well-known deductive engines for first-order logic. Compared with the direct approach of finding a proof algorithm for a specific class of modal logics, the translation methods have the advantage of being *independent* of the particular modal logic under consideration: a single theorem prover may be used for any translatable modal logic. Unfortunately, most of the proposed methods can be directly applied only to first-order complete modal logics, i.e. modal logics whose Kripke semantics is first-order axiomatizable. Many efforts have been devoted to deal with more challenging (widely used) modal logics. As a matter of fact, almost all the translations in the literature deal with specific classes of modal logics in an ad-hoc manner, and work only for the propositional case. (It must be also noticed that the application of some form of translation to first-order modal logics must be able to cope with the variety of ways to lift a given propositional modal logic to the first-order level proposed in the literature.)

In [DMP95], D'Agostino et al. proposed a novel method for translating modal logics into simple set theories. The main feature of such a method is that the use of a set theory (a suitable fragment of second-order logic) as target formalism allows a uniform translation of any normal, finitely axiomatizable (poly)modal logic. Unlike the standard relational translation and its variations [BD83, Oh93, No93], the set-theoretic translation works for all complete modal logics, re-

gardless of the first-order axiomatizatibility of their semantics. Furthermore, it also works if the modal logic under consideration is specified only by Hilbert axioms. Finally, it can be applied to extended modal logics by tuning the set theory driving the translation in a suitable way.

A new method for automated reasoning in modal logic (as well as in other nonclassical logics), that presents some similarities with the set-theoretic translation method, has been recently proposed in [Oh98]. It combines the (standard) translation approach and Hilbert style reasoning in a resolution framework, in order to deal with modal logics with a second-order semantics and/or specified by Hilbert axioms only.

In this paper, we present an elegant extension of the set-theoretic translation method for propositional modal logics to an important class of first-order modal logics (by far the most used), namely, objectual logics with rigid designators and fixed domain.

The rest of the paper is organized as follows. In Section 2 we present the basic features of the set-theoretic translation for propositional modal logics. In Section 3 we review basic definitions and results about first-order modal logics. In Section 4 we show how to extend the set-theoretic translation to first-order modal logics.

2 THE $\Box$-AS-*Pow* TRANSLATION

In this section, we describe the set-theoretic translation technique applied to the base case of K, prove its faithfulness, and show how to deal with modal logics extending this minimal system. An up-to-date presentation of the set-theoretic translation for *propositional* modal logics can be found in [DMP99].

We will use a fairly standard syntax for propositional modal logic, consisting of propositional variables (or letters) $P_1, P_2, \ldots$, logical connectives $\vee, \neg$, and the modal operator $\Box$. Derived symbols, to be used as abbreviations, are $\Diamond$ (defined as $\neg\Box\neg$), $\wedge, \rightarrow$, and $\leftrightarrow$. Well-formed formulas are defined as usual, exploiting the $\Box$ operator as a unary operator.

The so-called *standard translation* of a modal formula $\theta(P_1, \ldots, P_n)$ is the first-order formula $ST(\theta)(x_0, \mathtt{P}_1, \ldots, \mathtt{P}_n)$, where $\mathtt{P}_1, \ldots, \mathtt{P}_n$ are set variables corresponding to propositional letters $P_1, \ldots, P_n$, defined as follows [Oh91]:

- $ST(P_i) = \mathtt{P}_i(x_0)$;
- $ST(\varphi \vee \psi) = ST(\varphi) \vee ST(\varphi)$;
- $ST(\neg\varphi) = \neg ST(\varphi)$;
- $ST(\Box\varphi) = \forall y\ (x_0 R y \rightarrow ST(\varphi)(y|x_0))$,

where y does not occur in $ST(\varphi)$ and $y|x_0$ denotes uniform substitution of y for x_0. The *closed standard translation* $\overline{ST}(\varphi)$ of a modal formula φ is defined as the second-order sentence $\forall \mathtt{P}_1 \ldots \forall \mathtt{P}_n \forall x_0\ ST(\varphi)$, and it is easy to see that

$$\psi \models_f \varphi \text{ if and only if } \overline{ST}(\psi) \models \overline{ST}(\varphi),$$

where $\models_f$ and $\models$ denote frames logical consequence and second-order logical consequence, respectively.

2.1 BASIC MODAL LOGICS

The basic idea of the set-theoretic translation is simply to replace the accessibility relation R of the Kripke semantics with the membership relation $\in$. A world v accessible from w becomes an *element* of w and a further step from v, using the accessibility relation R, will amount to *looking into* v in order to reach for one of its elements. Other interesting consequences are that worlds, frames, and valuations of propositional variables become simply *sets* (of worlds), and that a frame $\mathcal{F}$ can be identified with its support W, being the accessibility relation implicitly defined as the membership relation on W.

Moreover, since we clearly want all worlds v accessible from a given world w in a frame W to turn out themselves to be elements of W, it is natural to require that all frames are *transitive* sets.

As a valuation for a propositional variable is nothing but a set of worlds, the standard definition of $\models$ will allow us to associate a set of worlds to each propositional formula. This set, inductively defined on the structural complexity of the formula, will be the collection of those worlds in the frame in which the formula holds. Let φ be a propositional modal formula built on the propositional variables $P_1, \ldots, P_n$, and let x be a variable used to denote the set W of all possible worlds. The set-theoretic translation $T_\varphi(x, y_1, \ldots, y_n)$ of φ is a term, inductively defined as follows:

$$\begin{aligned}
T_{P_i}(x, y_1, \ldots, y_n) &= y_i; \\
T_{\varphi\vee\psi}(x, y_1, \ldots, y_n) &= T_\varphi(x, y_1, \ldots, y_n) \cup \\
&\quad T_\psi(x, y_1, \ldots, y_n); \\
T_{\neg\varphi}(x, y_1, \ldots, y_n) &= x \setminus T_\varphi(x, y_1, \ldots, y_n); \\
T_{\Box\varphi}(x, y_1, \ldots, y_n) &= Pow(T_\varphi(x, y_1, \ldots, y_n)).
\end{aligned}$$

The term $T_{P_i}(x, y_1, \ldots, y_n)$ denotes the set of worlds at which the propositional variable P_i holds; the terms $T_{\varphi \vee \psi}(x, y_1, \ldots, y_n)$ and $T_{\neg\varphi}(x, y_1, \ldots, y_n)$ are obtained by means of the familiar rules for propositional connectives; the term $T_{\Box\varphi}(x, y_1, \ldots, y_n)$ is entirely forced by the Kripke semantics of $\Box$ together with the assumption that the accessibility relation R is replaced by $\in$.

If we let the y_i's and x be (pairwise distinct) set variables, $T_\varphi(x, y_1, \ldots, y_n)$ ($T_\varphi(x, \vec{y})$, from now on) is a syntactic translation mapping modal formulas into set-theoretic terms written in a language endowed with the symbols $\cup, \setminus$, and *Pow*. From now on such a translation will be called $\Box$-*as-Pow translation* ("box-as-powerset" translation).

By applying the $\Box$-as-*Pow* translation, the fact that a formula φ is satisfied in the frame W, with respect to a given valuation $P_1, \ldots, P_n$, amounts to saying that the substitution of W for x and of P_i for y_i satisfies $x \subseteq T_\varphi(x, \vec{y})$. To say that a formula φ is satisfied in W corresponds to saying that $\forall \vec{y}(x \subseteq T_\varphi(x, \vec{y}))$. Finally, the fact that φ is valid is stated as $\forall x, \vec{y}(x \subseteq T_\varphi(x, \vec{y}))$.

The next step is to provide a system of set-theoretic axioms which allows one to use the $\Box$-as-*Pow* translation to prove modal theorems. In order to isolate the principles governing the membership relation in this area, the first fact that must be taken into account is that $\in$ cannot have properties that cannot be guaranteed also for a generic accessibility relation R. Hence, $\in$ can be neither acyclic nor extensional, and a "minimalist" approach to axiomatic set theory becomes a *necessity* in this context. The axiomatic set theory that was introduced for this purpose (called Ω) is extremely simple: its axioms are, essentially, the definitions of the set-theoretic operators employed in the $\Box$-as-*Pow* translation.

$$\begin{aligned} x \in y \cup z &\leftrightarrow x \in y \vee x \in z; \\ x \in y \setminus z &\leftrightarrow x \in y \wedge x \notin z; \\ x \subseteq y &\leftrightarrow \forall z(z \in x \rightarrow z \in y); \\ x \in Pow(y) &\leftrightarrow x \subseteq y. \end{aligned}$$

The key feature of the theory Ω is its (*quasi*) minimality in providing a formal counterpart to the idea underlying the $\Box$-as-*Pow* translation. This fact is expressed by the following two results:

$$\begin{aligned} \vdash_{\mathsf{K}} \varphi &\Rightarrow \Omega \vdash \forall x\big(Trans(x) \rightarrow \forall \vec{y}(x \subseteq T_\varphi(x, \vec{y}))\big), \\ \models_f \varphi &\Leftarrow \Omega \vdash \forall x\big(Trans(x) \rightarrow \forall \vec{y}(x \subseteq T_\varphi(x, \vec{y}))\big). \end{aligned}$$

where $Trans(x)$ stands for $\forall y(y \in x \rightarrow y \subseteq x)$ (transitivity of x).

The $\Box$-as-*Pow* translation can be applied to any normal finitely axiomatizable modal logic, possibly specified by Hilbert axioms only. Let ψ be an axiom schema and let $\mathbf{L}$ be the modal logic obtained by adding ψ to K. The above results become:

$$\begin{aligned} \vdash_{\mathbf{L}} \varphi &\Rightarrow \Omega \vdash \forall x\big(Trans(x) \wedge \\ &\quad \forall \vec{z}(x \subseteq T_\psi(x, \vec{z})) \rightarrow \forall \vec{y}(x \subseteq T_\varphi(x, \vec{y}))\big), \\ \psi \models_f \varphi &\Leftarrow \Omega \vdash \forall x\big(Trans(x) \wedge \\ &\quad \forall \vec{z}(x \subseteq T_\psi(x, \vec{z})) \rightarrow \forall \vec{y}(x \subseteq T_\varphi(x, \vec{y}))\big). \end{aligned}$$

Moreover, if $\mathbf{L}$ turns out to be frame-complete (cf. [Ben85]), the two above results guarantee an exact correspondence between modal and set-theoretic derivability. In such cases, the following holds:

$$\begin{aligned} \vdash_{\mathbf{L}} \varphi &\Leftrightarrow \Omega \vdash \forall x\big(Trans(x) \wedge \\ &\quad \forall \vec{z}(x \subseteq T_\psi(x, \vec{z})) \rightarrow (\forall \vec{y}(x \subseteq T_\varphi(x, \vec{y}))\big). \end{aligned}$$

The above formulation suggests another possible reading of the effect of the $\Box$-as-*Pow* translation: when applied, it provides a form of deduction theorem for modal logics *modulo* the underlying set theory Ω.

As a matter of fact, Ω is not the minimal set theory capable of driving the $\Box$-as-*Pow* translation. The weaker theory MM ("Minimal Modal") obtained by replacing the axiom defining *Pow* by the following two theorems of Ω:

$$\begin{aligned} x \subseteq y &\rightarrow Pow(x) \subseteq Pow(y); \\ Pow(x \cap y) &\subseteq Pow(x) \cap Pow(y), \end{aligned}$$

is the minimal set theory that can be used to drive the translation.

The faithfulness of the $\Box$-as-*Pow* translation was originally proved in [DMP95] by using tools of non-well-founded set theory. The proof consists of two steps. The first step shows that, whenever a modal formula φ is derivable from a modal formula ψ in K, the theory Ω derives the translating sentence relating T_ψ and T_φ (*completeness* of the translation). The second step shows the *soundness* of the translation, namely, that whenever Ω proves the translating sentence relating T_ψ and T_φ, the formula φ is a frame logical-consequence of ψ.

Theorem 1 (Completeness of $\Box$-as-*Pow*)

$$\begin{aligned} \vdash_{\mathbf{L}} \varphi &\Rightarrow \Omega \vdash \forall x\big(Trans(x) \wedge \\ &\quad \forall \vec{z}(x \subseteq T_\psi(x, \vec{z})) \rightarrow \forall \vec{y}(x \subseteq T_\varphi(x, \vec{y}))\big). \end{aligned}$$

The completeness proof is by induction on the length of the derivation $\vdash_{\mathbf{L}} \varphi$. It is rather straightforward,

except for some care to be taken in order to cope with the lack of extensionality in Ω. As an example, the commutativity of set union $A \cup B = B \cup A$ can not be assumed in models of Ω. Nevertheless, the proof can be carried out on the weaker condition that $A \cup B \subseteq B \cup A \wedge B \cup A \subseteq A \cup B$, which holds in every model of Ω.

Theorem 2 (Soundness of $\Box$-as-*Pow*)

$$\psi \models_f \varphi \quad \Leftarrow \quad \Omega \vdash \forall x\big(Trans(x) \wedge \forall \vec{z}\,(\, x \subseteq T_\psi(x,\vec{z})\,) \rightarrow \forall \vec{y}\,(\, x \subseteq T_\varphi(x,\vec{y})\,)\big).$$

For any given frame (W, R), the soundness proof essentially builds a corresponding Ω-model, such that the frame is a transitive element x of the model and R is mimicked by $\in$. Since R can contain "cycles", we need a non-well-founded universe. The Ω-model can be built in such a way that, for all modal formulas φ, φ is valid in (W, R) if and only if $\forall \vec{z}(x \subseteq T_\varphi(x, \vec{z}))$ holds in the model (cf. [DMP95]).

An alternative proof of the (completeness and) soundness of the $\Box$-as-*Pow* translation, based on a comparison between the standard translation and the $\Box$-as-*Pow* translation, was given in [BDMP97], where van Benthem et al. also showed that the $\Box$-as-*Pow* translation can be adapted to deal with the so-called *general frame semantics*, thus capturing full K-derivability.

2.2 POLYMODAL AND EXTENDED MODAL LOGICS

The $\Box$-as-*Pow* translation for propositional modal logics has been extended to manage polymodal and extended modal logics.

In [DMP95], D'Agostino et al. showed how to generalize the $\Box$-as-*Pow* translation to polymodal logics. The main problem they dealt with was to map a polymodal frame, consisting of a set U endowed with k accessibility relations $\lhd_1, \ldots, \lhd_k$, with $k > 1$, into a set provided with the membership relation only. They solved this problem by preliminary providing polymodal logics with an alternative semantics that transforms the plurality of accessibility relations $\lhd_1, \ldots, \lhd_k$ into a single accessibility relation R together with k subsets $U_1, \ldots, U_k$ of U. In [MP97b], Montanari and Policriti showed how to adapt the $\Box$-as-*Pow* translation for polymodal logics with finitely many modalities to encompass the infinite number of accessibility relations of graded modal logics. In [MP97a], they showed that the $\Box$-as-*Pow* translation can also be applied to metric temporal logics. As a matter of fact, graded modal logics and metric temporal logics are treated as a special case of a more general technique able to deal with polymodal logics with infinitely many modalities.

In [BDMP98], on the ground of the translation for *general frame semantics*, van Benthem et al. showed that playing with the underlying axiomatic set-theory, a number of extended modal logics can be translated into set theory along the lines of the $\Box$-as-*Pow* translation, ranging from minimal tense logic to the weak system of second-order logic called L_2. The set theories underlying the translations are theories whose axioms are closely related to Gödel operations defining the constructible universe.

Both generalizations (polymodal and extended modal logics) were done *within* the propositional framework. In the following, we will show how to lift the set-theoretic translation to (a widely used class of) first-order modal logics.

3 FIRST-ORDER MODAL LOGICS

The extension of modal logics to the first-order case is not unique and gives rise to a large number of variants, which are classified and analyzed by Garson in [Ga84]. We concentrate our attention on one class of them, namely, the ones Garson calls **Q1** systems, i.e. objectual logics with rigid designators and fixed domain. They are far the most used in computer science and artificial intelligence, especially in their temporal form.

The following definitions and results are taken from the already cited [Ga84] and from Fitting's survey [Fi90].

Definition 3 The *language* of a first-order modal logic is given by:

- an infinite set of *variables*, denoted by x_1, x_2, etc.;
- a set of *constant symbols*, a set of *functional symbols*, and a set of *relational symbols*, denoted by P, Q, R and indexed variants of these;
- the *boolean connectives* $\vee$ and $\neg$;
- the *modal operator* $\Box$;
- the *universal quantifier* $\forall$;
- the *auxiliary symbols* (and).

The structure of formulas is the obvious one, and thus we do not formally define it. Moreover, we keep the right to use any derived connective, with its usual meaning, without further notice.

Let **L** be a propositional modal logic[1]. The *semantics* of the **Q1-L** system of quantified modal logic with rigid designators and fixed domain based on **L** is defined as follows.

Definition 4 A (**Q1**) *frame* is a triple $\mathcal{F} = (\mathcal{W}, R, \mathcal{D})$, where $\mathcal{W}$ is a non empty set of *worlds*, $R \subseteq \mathcal{W} \times \mathcal{W}$ is the *accessibility (reachability) relation*, and $\mathcal{D} \neq \emptyset$ is the *domain of interpretation*. A *(rigid) interpretation* on $\mathcal{F}$ is a triple $(\mathcal{F}_C, \mathcal{F}_F, \mathcal{F}_R)$, where:

- $\mathcal{F}_C$ assigns an element $\mathcal{F}_C(c)$ of $\mathcal{D}$ to each constant symbol c;
- $\mathcal{F}_F$ assigns a function $\mathcal{F}_F(f) : \mathcal{D}^k \to \mathcal{D}$ to each k-ary functional symbol f;
- $\mathcal{F}_R$ assigns a subset $\mathcal{F}_R(Q, a)$ of $\mathcal{D}^k$ to each k-ary relational symbol Q and world $a \in \mathcal{W}$.

A (**Q1**) *model* is a 6-tuple $\mathcal{M} = (\mathcal{W}, R, \mathcal{D}, \mathcal{F}_C, \mathcal{F}_F, \mathcal{F}_R)$ such that $\mathcal{F} = (\mathcal{W}, R, \mathcal{D})$ is a (**Q1**) frame and $(\mathcal{F}_C, \mathcal{F}_F, \mathcal{F}_R)$ is an interpretation on $\mathcal{F}$. We say that $\mathcal{M}$ is *based on* $\mathcal{F}$.

An *assignment* σ is a function mapping each variable x_i to an element $\sigma(x_i)$ of $\mathcal{D}$. We define $\sigma(s)$ for a term s and truth/validity of a formula in the obvious ways. Note that $\sigma(s)$ does not depend on a world — this is the meaning of the words *rigid interpretation*. $\mathcal{F}$ is a **Q1-L** *frame* if it validates all the formulas of **L**; $\mathcal{M}$ is a **Q1-L** *model* if it is based on a **Q1-L** frame. We use the symbols

$$a, \sigma \models_{\mathbf{Q1\text{-}L}} \varphi$$

to denote the truth of a formula φ in a world a under interpretation σ,

$$\mathcal{M} \models_{\mathbf{Q1-L}} \varphi$$

to denote its truth in a model $\mathcal{M}$, and

$$\models_{\mathbf{Q1\text{-}L}} \varphi$$

to denote its validity in all the **Q1-L** frames, and so on.

Definition 5 Let **L** be a propositional modal logic characterized by a Hilbert system. The following defines the **Q1-L** Hilbert calculus:

Propositional Axioms Any formula of the first-order modal language that is an instance of a theorem of the propositional modal logic **L** is an axiom.

[1]We view a logic as a set of formulas — the set of its theorems. For a full insight in this approach, see [BS84].

Universal Quantifier Axioms $\forall x_i \psi(x_i) \to \psi(s)$ is an axiom provided the substitution of s for x_i in $\psi(x_i)$ is free for x_i.

Modus Ponens $\dfrac{\varphi \quad \varphi \to \psi}{\psi}$.

Necessitation Rule $\dfrac{\varphi}{\Box\varphi}$.

Universal Generalization Rule $\dfrac{\varphi \to \psi}{\varphi \to \forall x_i \psi}$, provided that x_i has no free occurrence in φ.

Barcan Formulas For any formula $\psi(x_i)$, the *Barcan Formula* $\forall x_i \Box\psi(x_i) \to \Box\forall x_i \psi(x_i)$ is an axiom.

For the rest of this paper, we will assume that **L** has a Hilbert system with only one modal axiom scheme, which we will usually call γ. In this case, we can take the following alternative definition of Propositional Axioms:

Propositional Axioms Any (first-order, modal) instance of a classical propositional tautology, K or γ is an axiom.

We shall write

$$\vdash_{\mathbf{Q1\text{-}L}} \varphi$$

to denote that φ is derivable in the **Q1-L** Hilbert calculus.

Theorem 6 *If* **L** *is a frame-complete propositional modal logic and* φ *is a first-order modal formula, then*

$$\vdash_{\mathbf{Q1\text{-}L}} \varphi \Leftrightarrow \models_{\mathbf{Q1\text{-}L}} \varphi.$$

4 □-as-*Pow* TRANSLATION FOR FULLY QUANTIFIED MODAL LOGICS

In this section, we show how the set-theoretic translation for the propositional case can be extended to **Q1** quantified modal systems. The target of the translation is a two-sorted variant of Ω that we call Ω_2. The method achieves full generality, i.e. we give soundness and completeness theorems working for any underlying frame-complete propositional modal logic.

Definition 7 Let $\mathcal{L}$ be the original first-order modal language. The *language* $\mathcal{L}''$ of the first-order set-theoretic translation is a two-sorted first-order language with sorts `set` and `term`, together with the following symbols:

- all the constant/function symbols of $\mathcal{L}$ are constant/function symbols of $\mathcal{L}''$, with the same arity and domains and co-domains **term**;
- for each k-ary relational symbol Q of $\mathcal{L}$, there is a k-ary functional symbol f_Q in $\mathcal{L}''$ of sort $\mathbf{term}^k \to \mathbf{set}$; moreover, for some formula φ we will introduce a corresponding function symbol f_φ of sort $\mathbf{term}^m \to \mathbf{set}$, where m is the number of free variables in φ (we assume all f_Q's and f_φ's to be distinct for different Q's and φ's);
- the functional symbols $\cup$, $\setminus$ (binary) and *Pow* (unary) and the binary relational symbols $\in$ and $\subseteq$ are in $\mathcal{L}''$; all their domain and co-domain sorts are **set**.

Definition 8 Ω_2 is the theory, on the language $\mathcal{L}''$, defined by the following axioms (all the free variables are intended to be of sort **set** and universally quantified):

$$\begin{aligned} x \in y \cup z &\leftrightarrow x \in y \vee x \in z; \\ x \in y \setminus z &\leftrightarrow x \in y \wedge x \notin z; \\ x \subseteq y &\leftrightarrow \forall z : \mathbf{set}\,(z \in x \to z \in y); \\ x \in \mathit{Pow}(y) &\leftrightarrow x \subseteq y. \end{aligned}$$

Definition 9 The *set-theoretic translation* of a formula φ is a term $t_\varphi(x, y_1, \ldots, y_n)$, inductively defined by the following equations, where $\{y_1, \ldots, y_n\}$ is a superset of the set of the variables appearing in φ:

$$\begin{aligned} t_\varphi(x, y_1, \ldots, y_n) &= f_Q(s_1, \ldots, s_k) \\ &\quad \text{if } \varphi = Q(s_1, \ldots, s_k), \\ t_\varphi(x, y_1, \ldots, y_n) &= f_\varphi(x_{j_1}, \ldots, x_{j_m}) \\ &\quad \text{if } \varphi = \forall x_i\, \eta \text{ and } x_{j_1}, \ldots, x_{j_m} \\ &\quad \text{are the free variables in } \varphi, \\ &\quad \text{with } j_1 < j_2 < \cdots < j_m, \\ t_\varphi(x, y_1, \ldots, y_n) &= t_\eta(x, y_1, \ldots, y_n) \cup \\ &\quad t_\theta(x, y_1, \ldots, y_n) \\ &\quad \text{if } \varphi = \eta \vee \theta, \\ t_\varphi(x, y_1, \ldots, y_n) &= x \setminus t_\eta(x, y_1, \ldots, y_n) \\ &\quad \text{if } \varphi = \neg\eta, \\ t_\varphi(x, y_1, \ldots, y_n) &= \mathit{Pow}(t_\eta(x, y_1, \ldots, y_n)) \\ &\quad \text{if } \varphi = \Box\eta. \end{aligned}$$

We also define the $\alpha(x)$ and $\beta(x)$ formulas as follows:

$$\alpha_\varphi(x) = \begin{cases} \forall y_{j_1}, \ldots, y_{j_m} : \mathbf{term}\, \forall y : \mathbf{set} \\ (y \in x \to (y \in f_\varphi(y_{j_1}, \ldots, y_{j_m}) \leftrightarrow \\ \forall y_i : \mathbf{term}\, y \in t_\eta(x, y_1, \ldots, y_n))) \\ \quad \text{if } \varphi = \forall y_i\, \eta \text{ and } y_{j_1}, \ldots, y_{j_m} \text{ are} \\ \quad \text{the free variables in } \varphi, \text{ with} \\ \quad j_1 < j_2 < \cdots < j_m, \\ \mathit{true} \text{ otherwise;} \end{cases}$$

$$\beta_\varphi(x) = \bigwedge_{\eta \le \varphi} \alpha_\eta(x),$$

where the conjunction is over all the subformulas η of φ.

If **L** is a propositional modal logic with characteristic axiom γ, built on propositional variables $y_1, \ldots, y_r$, we define $\mathit{Axiom}^2_{\mathbf{L}}(x)$ as the formula

$$\forall y_1, \ldots, y_r : \mathbf{set}\; x \subseteq T_\gamma(x, y_1, \ldots, y_r),$$

where T_γ is defined exactly as in the propositional case.

$\mathit{Trans}(x)$ stands for $x \subseteq \mathit{Pow}(x)$ as usual.

The intuition behind Definition 9 is that $t_\varphi(x, d_1, \ldots, d_n)$ represents the set of worlds in which φ is true under the interpretation $[y_1/d_1, \ldots, y_n/d_n]$ (the parameter x has the same relativizing meaning as in the propositional translation). The α_φ will be used as additional hypotheses to define the meaning of $t_{\forall y_i \eta}$ in terms of t_η. There is no way to do this in a general fashion, e. g. by means of an additional axiom in Ω_2, because such an axiom would have to deal with variable names (not values), which is impossible in first-order logic.

Lemma 10 *If there is no free occurrence of y_i in φ, then there is no occurrence of y_i in $t_\varphi(x, y_1, \ldots, y_n)$.*

Proof. Directly from Definition 9. ∎

Theorem 11

$$\begin{aligned} \vdash_{\mathbf{Q1\text{-}L}} \varphi \;\Rightarrow\; & \Omega_2 \vdash \forall x : \mathbf{set}\,(\mathit{Trans}(x) \wedge \\ & \mathit{Axiom}^2_{\mathbf{L}}(x) \wedge \beta_\varphi(x) \to \forall y_1, \ldots, \\ & y_n : \mathbf{term}\; x \subseteq t_\varphi(x, y_1, \ldots, y_n)). \end{aligned}$$

Proof. The proof is by induction on the length of the derivation $\vdash_{\mathbf{Q1-L}} \varphi$.

Propositional Axioms. The proof is exactly the same as the one for the propositional set-theoretic translation (cf. [DMP95]).

Universal Quantifier Axioms. Let us assume, without loss of generality, that we are instantiating the variable x_1 and the other free variables in $\psi(x_1)$ are $x_2, \ldots, x_m$, $m \le n$. Thus φ is

$$\forall x_1\, \psi(x_1) \to \psi(s),$$

where the substitution $\{x_1 \mapsto s\}$ is free for x_1 in $\psi(x_1)$. Let η be $\forall x_1\, \psi(x_1)$. Then, by Definition 9,

$$\begin{aligned} t_\varphi(x, x_1, \ldots, x_n) &= (x \setminus f_\eta(x_2, \ldots, x_m)) \cup \\ &\quad t_{\psi(s)}(x, x_1, \ldots, x_n) \end{aligned}$$

and

$$\begin{aligned}\alpha_\eta(x) &= \forall x_2,\ldots,x_m : \mathtt{term}\,\forall y : \mathtt{set}\\ &\quad (y \in x \to (y \in f_\eta(x_2,\ldots,x_m) \leftrightarrow\\ &\quad \forall x_1 : \mathtt{term}\, y \in t_{\psi(x_1)}(x,x_1,\ldots,x_n))),\\ \beta_\varphi(x) &= \alpha_\eta(x) \wedge \beta_{\psi(x_1)}(x).\end{aligned}$$

Assume $\beta_\varphi(x)$ ($Trans(x)$ and $Axiom^2_{\mathbf{L}}(x)$ are not needed in the proof). We shall prove that $x \subseteq t_\varphi(x,x_1,\ldots,x_n)$ by showing that, whenever $y \in x$ and $y \notin t_{\psi(s)}(x,x_1,\ldots,x_n)$, $y \notin f_\eta(x_2,\ldots,x_m)$. Assume $y \in x$ and $y \notin t_{\psi(s)}(x,x_1,\ldots,x_n)$. From Definition 9, we have that $t_{\psi(s)}(x,x_1,\ldots,x_n) = t_{\psi(x_1)}(x,x_1,\ldots,x_n)\{x_1 \mapsto s\}$; hence $\exists x_1 : \mathtt{term}\, y \notin t_{\psi(x_1)}(x,x_1,\ldots,x_n)$, that is, $\neg\forall x_1 : \mathtt{term}\, y \in t_{\psi(x_1)}(x,x_1,\ldots,x_n)$, which, together with $\beta_\varphi(x)$ ($\alpha_\eta(x)$ conjunct), implies $y \notin f_\eta(x_2,\ldots,x_m)$.

Modus Ponens and *Necessitation Rule.* Both proofs are basically the same as those for the propositional case. Refer, again, to [DMP95].

Universal Generalization Rule. Assume the same notation as with the Universal Quantifier Axiom. The rule has then the form

$$\frac{\varphi \to \psi}{\varphi \to \forall x_1\, \psi},$$

where x_1 has no free occurrence in φ. Let η be the subformula $\forall x_1\, \psi$. From Definition 9, we get

$$\begin{aligned}t_{\varphi\to\psi}(x,x_1,\ldots,x_n) &= (x \setminus t_\varphi(x,x_1,\ldots,x_n)) \cup\\ &\quad t_\psi(x,x_1,\ldots,x_n)\end{aligned}$$

$$\begin{aligned}t_{\varphi\to\forall x_1\,\psi}(x,x_1,\ldots,x_n) &= (x \setminus t_\varphi(x,x_1,\ldots,x_n)) \cup\\ &\quad f_\eta(x_2,\ldots,x_m),\end{aligned}$$

$$\begin{aligned}\alpha_\eta(x) &= \forall x_2,\ldots,x_m : \mathtt{term}\,\forall y : \mathtt{set}\\ &\quad (y \in x \to (y \in f_\eta(x_2,\ldots,x_m) \leftrightarrow\\ &\quad \forall x_1 : \mathtt{term}\, y \in t_\psi(x,x_1,\ldots,x_n))),\end{aligned}$$

$$\beta_{\varphi\to\forall x_1\,\psi}(x) = \alpha_\eta(x) \wedge \beta_\varphi(x) \wedge \beta_\psi(x).$$

Assume $\beta_{\varphi\to\forall x_1\,\psi}(x)$ ($Trans(x)$ and $Axiom^2_{\mathbf{L}}(x)$ are not needed in the proof). We shall prove that $x \subseteq t_{\varphi\to\forall x_1\,\psi}(x,x_1,\ldots,x_n)$ by showing that, whenever $y \in x$ and $y \in t_\varphi(x,x_1,\ldots,x_n)$, necessarily $y \in f_\eta(x_2,\ldots,x_m)$. Assume $y \in x$ and $y \in t_\varphi(x,x_1,\ldots,x_n)$. By induction hypothesis, $x \subseteq (x \setminus t_\varphi(x,x_1,\ldots,x_n)) \cup t_\psi(x,x_1,\ldots,x_n)$, hence $y \in t_\psi(x,x_1,\ldots,x_n)$. By Lemma 10, formula $y \in t_\varphi(x,x_1,\ldots,x_n)$ has no occurrence of x_1; hence, we can apply classical Universal Generalization to conclude that $\forall x_1 : \mathtt{term}\, y \in t_\psi(x,x_1,\ldots,x_n)$ and, thus, from conjunct $\alpha_\eta(x)$ of assumption $\beta_{\varphi\to\forall x_1\,\psi}(x)$, that $y \in f_\eta(x_2,\ldots,x_m)$.

Barcan Formulas. Let $\varphi = \forall y_1\, \Box\psi(y_1) \to \Box\forall y_1\, \psi(y_1)$. Let us call η and θ the formulas $\forall y_1\, \Box\psi(y_1)$ and $\forall y_1\, \psi(y_1)$, respectively, and let $y_2,\ldots,y_m$ be the free variables in η (or, equivalently, in θ). Then

$$\begin{aligned}t_\varphi(x,y_1,\ldots,y_n) &= (x \setminus f_\eta(y_2,\ldots,y_m)) \cup\\ &\quad Pow(f_\theta(y_2,\ldots,y_m)),\end{aligned}$$

$$\begin{aligned}\alpha_\eta(x) &= \forall y_2,\ldots,y_m : \mathtt{term}\,\forall y : \mathtt{set}\\ &\quad (y \in x \to (y \in f_\eta(y_2,\ldots,y_m) \leftrightarrow\\ &\quad \forall y_1 : \mathtt{term} y \in Pow(t_{\psi(y_1)}(x,y_1,\ldots,y_n))),\end{aligned}$$

$$\begin{aligned}\alpha_\theta(x) &= \forall y_2,\ldots,y_m : \mathtt{term}\,\forall y : \mathtt{set}\\ &\quad (y \in x \to (y \in f_\theta(y_2,\ldots,y_m) \leftrightarrow\\ &\quad \forall y_1 : \mathtt{term} y \in t_{\psi(y_1)}(x,y_1,\ldots,y_n))),\end{aligned}$$

$$\beta_\varphi(x) = \alpha_\eta(x) \wedge \alpha_\theta(x) \wedge \beta_{\psi(y_1)}(x).$$

Assume $\beta_\varphi(x)$ and $y \in x$. Then

$$\begin{aligned}&y \in t_\varphi(x,y_1,\ldots,y_n) \quad \leftrightarrow\\ &(y \in x \wedge y \notin f_\eta(y_2,\ldots,y_m)) \vee y \subseteq f_\theta(y_2,\ldots,y_m) \quad \leftrightarrow\\ &\exists y_1 : \mathtt{term} y \not\subseteq t_{\psi(y_1)}(x,y_1,\ldots,y_n) \vee\\ &\forall z : \mathtt{set}\,(z \in y \to z \in f_\theta(y_2,\ldots,y_m)) \quad \leftrightarrow\\ &\exists y_1 : \mathtt{term}\,\exists z : \mathtt{set}\,(z \in y \wedge\\ &z \notin t_{\psi(y_1)}(x,y_1,\ldots,y_n)) \vee \forall z : \mathtt{set}\\ &(z \in y \to \forall y_1 : \mathtt{term} z \in t_{\psi(y_1)}(x,y_1,\ldots,y_n)).\end{aligned}$$

The first equivalence is just the unfolding of the definition of $t_\varphi(x,y_1,\ldots,y_n)$ by the axioms of Ω_2. In the second we dropped the subformula $y \in x$ from the conjunct because it is true by assumption, then we rewrote the first disjunct using $\alpha_\eta(x)$ and the second using the Ω_2 axiom for $\subseteq$. In the last, we used the definition of $\subseteq$ in the left disjunct and $\alpha_\theta(x)$ in the right one.
Now, to show that $x \subseteq t_\varphi(x,y_1,\ldots,y_n)$, we take, as above, $y \in x$ and assume that the second disjunct of the last line is false, which is equivalent to

$$\exists z : \mathtt{set}\,(z \in y \wedge \exists y_1 : \mathtt{term} z \notin t_{\psi(y_1)}(x,y_1,\ldots,y_n)).$$

This obviously implies the first disjunct, thus proving the claim. ■

Theorem 12

$$\begin{aligned}&\Omega_2 \models \forall x : \mathtt{set}\,(Trans(x) \wedge\\ &Axiom^2_{\mathbf{L}}(x) \wedge \beta_\varphi(x) \to \forall y_1,\ldots,\\ &y_n : \mathtt{term} x \subseteq t_\varphi(x,y_1,\ldots,y_n)) \quad \Rightarrow \quad \models_{\mathbf{Q1\text{-}L}} \varphi.\end{aligned}$$

Proof. The proof of the above theorem is an adaptation of the technique introduced in the soundness proof of the propositional case in [DMP95]. More precisely, we are going to follow the proof of Theorem 5 of [DMP95]. In particular, the first part of the proof will almost be the same and we shall state some lemmas without proving them.

Let $\mathcal{U}$ be a universe of hypersets satisfying all the axioms of $ZF-FA$ (ZF except the Foundation Axiom) and AFA (see [Acz88] for details).

Lemma 13 *Let α be an ordinal, V_α be the set of all well-founded sets of rank less than α, and $\mathcal{U} \setminus V_\alpha$ be the universe of all hypersets not belonging to V_α. Then any model for the language of Ω_2 with domain $\mathcal{U} \setminus V_\alpha$ for sort* **set** *and interpretation function $(\cdot)'$ satisfiying*

$$\begin{array}{rcl} x \in' y & \textit{iff} & x \in y, \\ x \cup' y & = & x \cup y, \\ x \subseteq' y & \textit{iff} & x \setminus V_\alpha \subseteq y, \\ x \setminus' y & = & \begin{cases} x \setminus y & \textit{if } x \setminus y \notin V_\alpha, \\ V_\alpha & \textit{otherwise}, \end{cases} \\ Pow'(x) & = & \{y | y \setminus V_\alpha \subseteq x\} \end{array}$$

is a model of Ω_2.

Proof. See Lemma 6 of [DMP95]. ∎

Given a frame $(\mathcal{W}, R, \mathcal{D})$, we want to embed it into the universe $\mathcal{U} \setminus V_\alpha$, for some suitable α. In [DMP95], D'Agostino et al. prove the existence of a labelled decoration * such that the following lemma holds.

Lemma 14 *For each $a, b \in \mathcal{W}$,*

1. *$a \neq b \Rightarrow a^* \neq b^*$,*
2. *$a^* \notin V_{\alpha+1}$ and $a^* \setminus V_{\alpha+1} = \{b^* | aRb\}$.*

Proof. See (ii) and (iii) in Lemma 7 of [DMP95]. ∎

Now consider an interpretation $(\mathcal{F}_C, \mathcal{F}_F, \mathcal{F}_R)$ over $(\mathcal{W}, R, \mathcal{D})$ and an assignment $\sigma : \{x_1, \ldots, x_n\} \to \mathcal{D}$. Let $\mathcal{M}$ be the **Q1** model $(\mathcal{W}, R, \mathcal{D}, \mathcal{F}_C, \mathcal{F}_F, \mathcal{F}_R)$.

A model for the language of Ω_2 can be obtained as follows. First, take the interpretation for **set** symbols as defined by the previous lemmas and let $\mathcal{D}$ be the domain for the sort **term**.

Then, interpret **term** constant and function symbols as in $\mathcal{F}_C$ and $\mathcal{F}_F$, respectively, and, for each k-ary relational symbol Q, let $f^*_Q(d_1, \ldots, d_k)$ be equal to

$$\{a^* | a \in \mathcal{W} \text{ and } a, [x_1/d_1, \ldots, x_k/d_k] \models_{\mathbf{Q1\text{-}L}} Q(x_1, \ldots, x_k)\}$$

if such a set is not empty, and to $V_{\alpha+1}$ otherwise.

Finally, for each universally quantified formula $\varphi = \forall x_i \eta$, with free variables $x_{j_1}, \ldots, x_{j_m}$, let $f^*_\varphi(d_1, \ldots, d_m)$ be equal to

$$\{a^* | a \in \mathcal{W} \text{ and } a, [x_{j_1}/d_1, \ldots, x_{j_m}/d_m] \models_{\mathbf{Q1\text{-}L}} \varphi\}$$

if such a set is not empty, and to $V_{\alpha+1}$ otherwise.

We call this model $\mathfrak{A}_\mathcal{M}$. Furthermore, let $\mathcal{W}^*$ be equal to $\{a^* | a \in \mathcal{W}\}$. $\mathcal{W}^* \notin V_{\alpha+1}$ because $V_{\alpha+1}$ is transitive and, for each $a \in \mathcal{W}$, $a^* \notin V_{\alpha+1}$ (Lemma 14).

Lemma 15 *It holds that:*

1. *for each $a \in \mathcal{W}$, $d_1, \ldots, d_n \in \mathcal{D}$ and formula φ,*

$$a, [x_1/d_1, \ldots, x_n/d_n] \models_{\mathbf{Q1\text{-}L}} \varphi \quad \textit{if and only if} \quad a^* \in' t^*_\varphi(\mathcal{W}^*, d_1, \ldots, d_n) \textit{ in } \mathfrak{A}_\mathcal{M},$$

2. *$\mathfrak{A}_\mathcal{M} \models \beta_\varphi(\mathcal{W}^*)$.*

Proof.

1. Let $\sigma = [x_1/d_1, \ldots, x_n/d_n]$. The proof is by induction on φ.
   - $\varphi = Q(s_1, \ldots, s_k)$ immediate from the definition of f^*_Q.
   - $\varphi = \eta \vee \theta$ $a, \sigma \models_{\mathbf{Q1\text{-}L}} \varphi$ *iff* $a, \sigma \models_{\mathbf{Q1\text{-}L}} \eta$ or $a, \sigma \models_{\mathbf{Q1\text{-}L}} \theta$ *iff* (by induction hypothesis) $a^* \in' t^*_\eta(\mathcal{W}^*, d_1, \ldots, d_n)$ or $a^* \in' t^*_\theta(\mathcal{W}^*, d_1, \ldots, d_n)$ *iff* (for $\mathfrak{A}_\mathcal{M}$ is a model of Ω_2) $a^* \in' t^*_\eta(\mathcal{W}^*, d_1, \ldots, d_n) \cup' t^*_\theta(\mathcal{W}^*, d_1, \ldots, d_n)$ *iff* (by definition of t_φ) $a^* \in' t^*_\varphi(\mathcal{W}^*, d_1, \ldots, d_n)$.
   - $\varphi = \neg\eta$ $a, \sigma \models_{\mathbf{Q1\text{-}L}} \varphi$ *iff* $a, \sigma \not\models_{\mathbf{Q1\text{-}L}} \eta$ *iff* (by i.h.) $a^* \notin' t^*_\eta(\mathcal{W}^*, d_1, \ldots, d_n)$ *iff* (for $a^* \in' \mathcal{W}^*$) $a^* \in' \mathcal{W}^*$ and $a^* \notin' t^*_\eta(\mathcal{W}^*, d_1, \ldots, d_n)$ *iff* (for $\mathfrak{A}_\mathcal{M}$ is a model of Ω_2) $a^* \in' \mathcal{W}^* \setminus t^*_\eta(\mathcal{W}^*, d_1, \ldots, d_n)$ *iff* (by definition of t_φ) $a^* \in' t^*_\varphi(\mathcal{W}^*, d_1, \ldots, d_n)$.
   - $\varphi = \Box\eta$ $a, \sigma \models_{\mathbf{Q1\text{-}L}} \varphi$ *iff* for every $b \in \mathcal{W}$ such that aRb, $b, \sigma \models_{\mathbf{Q1\text{-}L}} \eta$ *iff* (by i.h.) for every $b \in \mathcal{W}$ such that aRb, $b^* \in' t^*_\eta(\mathcal{W}^*, d_1, \ldots, d_n)$ *iff* (by Lemma 14) $a^* \setminus V_{\alpha+1} \subseteq t^*_\eta(\mathcal{W}^*, d_1, \ldots, d_n)$ *iff* (by definition of Pow' in Lemma 13) $a^* \in' Pow(t^*_\eta(\mathcal{W}^*, d_1, \ldots, d_n))$ *iff* (by definition of t_φ) $a^* \in' t^*_\varphi(\mathcal{W}^*, d_1, \ldots, d_n)$.

$\varphi = \forall x_i \eta$ immediate from the definitions of f^*_φ and t_φ.

2. Assume, without loss of generality, that $\varphi = \forall x_1 \eta$ and let $x_2, \ldots, x_m$ be the free variables in φ. We are going to prove $\alpha_\varphi(\mathcal{W}^*)$. Let $a \in \mathcal{W}$ and $d_1, \ldots, d_n$ be arbitrary elements of $\mathcal{D}$. Then $a^* \in' f^*_\varphi(d_2, \ldots, d_m)$ *iff* (by definition of f^*_φ) $a, [x_2/d_2, \ldots, x_m/d_m] \models_{\mathbf{Q1\text{-}L}} \varphi$ *iff* for every $d_1 \in \mathcal{D}$, $a, [x_1/d_1, \ldots, x_m/d_m] \models_{\mathbf{Q1\text{-}L}} \eta$ *iff* (for the first part of this lemma) for every $d_1 \in \mathcal{D}$, $a^* \in' t^*_\eta(\mathcal{W}^*, d_1, \ldots, d_n)$[2] *iff* $\forall x_1 : \mathbf{term} a^* \in' t^*_\eta(\mathcal{W}, x_1, d_2, \ldots, d_n)$.

■

Corollary 16

$$\mathcal{M} \models_{\mathbf{Q1\text{-}L}} \varphi$$

if and only if

$$\forall x_1, \ldots, x_n : \mathbf{term}\, \mathcal{W}^* \subseteq' t^*_\varphi(\mathcal{W}^*, x_1, \ldots, x_n) \text{ in } \mathfrak{A}_\mathcal{M}.$$

Proof. Follows directly from Lemma 15. Remember that $\mathcal{M} \models_{\mathbf{Q1\text{-}L}} \varphi$ means $\mathcal{M} \models_{\mathbf{Q1\text{-}L}} \forall x_1, \ldots, x_n \varphi$.

■

Lemma 17 *A modal propositional formula γ on propositional variables $P_1, \ldots, P_r$ is valid in the frame $(\mathcal{W}, R)$ if and only if, for the corresponding hyperset $\mathcal{W}^*$,*

$$\forall x_1, \ldots, x_r : \mathbf{set}\, \mathcal{W}^* \subseteq' t^*_\gamma(\mathcal{W}^*, x_1, \ldots, x_r)$$

holds in $\mathcal{U} \setminus V_{\alpha+1}$.

Proof. See Lemma 9 of [DMP95]. ■

To conclude the proof of Theorem 12, let us suppose that

$$\Omega_2 \models \forall x : \mathbf{set}\, (Trans(x) \wedge Axiom^2_\mathbf{L}(x) \wedge \beta_\varphi(x) \rightarrow \forall x_1, \ldots, x_n : \mathbf{term}\, x \subseteq t_\varphi(x, x_1, \ldots, x_n)).$$

Let $\mathcal{M} = (\mathcal{W}, R, \mathcal{D}, \mathcal{F}_C, \mathcal{F}_F, \mathcal{F}_R)$ be a model based on a frame in which γ is valid (a **Q1-L** model). From Lemma 17 it follows that $Axiom^2_\mathbf{L}(\mathcal{W}^*)$ is true in $\mathfrak{A}_\mathcal{M}$, and from the second part of Lemma 15 the same follows for $\beta_\varphi(\mathcal{W})$. Furthermore, it is easy to prove that $Trans(\mathcal{W}^*)$ holds as well. Since $\mathfrak{A}_\mathcal{M}$ is a model of Ω_2 and from the hypothesis, the formula $\forall x_1, \ldots, x_n : \mathbf{term}\, \mathcal{W}^* \subseteq' t^*_\varphi(\mathcal{W}^*, x_1, \ldots, x_n)$ is true in $\mathfrak{A}_\mathcal{M}$, which, by Corollary 16, implies that $\mathcal{M} \models_{\mathbf{Q1\text{-}L}} \varphi$.

■

Remark. Note that we state no equivalent for Lemma 9 of [DMP95] (Lemma 17) for quantified logics: in the proof of soundness for the propositional case (Theorem 2), that lemma is really needed only to infer the truth of $Axiom_\mathbf{L}(x)$ in $\mathcal{U} \setminus V_{\alpha+1}$ (that is how we use it in our proof, too); for the last step, Lemma 8 would be sufficient.

References

[Acz88] P. Aczel. *Non-Well-Founded Sets.* Center for the Studies of Language and Information, Stanford University, 1988.

[BD83] J. van Benthem and K. Doets. Higher-Order Logic. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume I, pages 275–329. D. Reidel, Dordrecht-Holland,1983.

[Ben85] J. van Benthem. *Modal Logic and Classical Logic.* Bibliopolis, Napoli and Atlantics Heights, NJ, 1985.

[BDMP97] J. van Benthem, G. D'Agostino, A. Montanari, and A. Policriti. Modal deduction in second-order logic and set theory-I. *Journal of Logic and Computation*, 7(2):251–265, 1997.

[BDMP98] J. van Benthem, G. D'Agostino, A. Montanari, and A. Policriti. Modal deduction in second-order logic and set theory-II. *Studia Logica*, 60(3):387–420, 1998.

[BS84] R. Bull and K. Segerberg. Basic Modal Logic. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume II, pages 1–88. D. Reidel, Dordrecht-Holland, 1984.

[DMP95] G. D'Agostino, A. Montanari, and A. Policriti. A set-theoretic translation method for poly-modal logics. *Journal of Automated Reasoning*, 15:317–337, 1995.

[DMP99] G. D'Agostino, A. Montanari, and A. Policriti. Modal Logic and Set Theory: a Set-Theoretic Interpretation of Modal Logic. In J. Gerbrandy, M. Marx, M. de Rijke, and Y. Venema, editors, *Liber Amicorum for the Fiftieth Birthday of Johan van Benthem*, ILLC, Amsterdam, NL, 1999.

[Fi90] M. Fitting. Basic Modal Logic. In D. M. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume I, pages 395–448. D. Reidel, Dordrecht-Holland, 1990.

[2] By Lemma 10, all the d's beyond the m-th do not count in the interpretation.

[Ga84] J. W. Garson Quantification in Modal Logic. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume II, pages 249–307. D. Reidel, Dordrecht-Holland, 1984.

[HdR95] W. van der Hoek and M. de Rijke. Counting Objects. *Journal of Logic and Computation*, 5(3):325–345, 1995.

[MP97b] A. Montanari and A. Policriti. A set-theoretic approach to automated deduction in graded modal logics. In *Proceedings of the International Joint Conference on Artificial Intellicence, IJCAI97*, pages 196–201. San Francisco: Morgan Kaufmann, 1997.

[MP97a] A. Montanari and A. Policriti. Executing metric temporal logic. In *Proceedings of the IJCAI97 Workshop on Programming in Temporal and Non Classical Logics*, Nagoya, Japan, 1997.

[No93] A. Nonnengart. First-Order Modal Logic Theorem Proving and Functional Simulation. In *Proceedings of the International Joint Conference on Artificial Intellicence, IJCAI93*, pages 80–85. San Francisco: Morgan Kaufmann, 1993.

[Oh91] H. J. Ohlbach. Semantic-Based Translation Methods for Modal Logics. *Journal of Logic and Computation*, 1(6):691–746, 1991.

[Oh93] H. J. Ohlbach. Translation Methods for Non-Classical Logics: An Overview. *The Logic Journal of IGPL*, 1(1):69–89, 1993.

[Oh98] H. J. Ohlbach. Combining Hilbert Style and Semantics Reasoning in a Resolution Framework. *Proceedings of the 15th International Conference on Automated Deduction, CADE-15*, LNCS 1421, Springer, 1998.

Temporal Reasoning II

Controllability characterization and checking in Contingent Temporal Constraint Networks

Thierry Vidal
LGP / ENIT
47, av d'Azereix - BP 1629
F-65016 Tarbes cedex, France
thierry@enit.fr

Abstract

Temporal Constraint Networks allow to express possible durations or delays between time-points, in the shape of intervals of values. A solution of such a network is a precise assignment of dates to time-points, hence fixing the duration for each constraint. If the agent can freely decide each effective duration, then she will know her problem has a solution by simply checking the well-known consistency property of the network. In many application areas though, as in planning and scheduling for instance, most of the durations are *contingent* in the sense that they will only be "observed" (or "received") by the agent, that needs now check a different kind of property: the *controllability* of the network. But what are the semantics of this new kind of knowledge ? How to model it ? Extending previous works on the topic, this paper gives a unified and more legible characterization of the different levels of controllabilities that can be defined. Then what reasoning methods can be designed to check them ? Focusing on the prominent one, the *Dynamic Controllability*, the paper gives a new complete checking method based on a translation of the Contingent Temporal Constraint Network into a *Timed Game Automaton*, that may furthermore be used for instance for execution control needs.

1 BACKGROUND AND OVERVIEW

Reified temporal logics differ from clasical ones in that the propositions are separated from their temporal qualifications [Vila and Reichgelt, 1993]. Queries (such as "is proposition P true at time t ?") are still processed at the logical level, but reasoning upon the mere temporal relations can be done through a purely temporal algebraic model. One usually gets a constraint network that is mainly used to check the temporal consistency of the given problem. Temporal Constraint Networks (TCN) [Schwalb and Dechter, 1997] rely on *qualitative* (or symbolic) constraint algebras [Allen, 1983, Vilain et al., 1989] but more specifically tackle *quantitative* (or numerical) constraints. They are now at the heart of many application domains such as scheduling [Dubois et al., 1993], supervision [Dousson et al., 1993], diagnosis and temporal databases [Brusoni et al., 1994], multimedia authoring sytems [Jourdan et al., 1997], or planning [Ghallab and Vidal, 1995, Morris et al., 1998].

Temporal Constraint Networks may be extended to take into account the inherent uncertain nature of durations of some tasks in realistic applications, distinguishing between *contingent* constraints (whose effective duration is only observed at execution time, e.g. the duration of a task) and *controllable* ones (which instantiation is controlled by the agent, e.g. a delay between starting times of tasks): the problem becomes a decision-making process under uncertainty, and consistency must be redefined in terms of *controllabilities*. The *Weak* one (i.e. existence of a solution for each "situation" likely to arise in the external world) and the *Strong* one (i.e. existence of one unique solution fitting all situations) have first [Vidal and Ghallab, 1996] been introduced, inspired by a similar distinction made in discrete CSPs [Fargier et al., 1996]. But to encompass the reactive nature of the solution building process in dynamic domains like planning, the *Dynamic controllability* property had to be added (i.e. existence of a solution built in the process of time, each assignment depending only on the situation observed so far): in [Vidal and Fargier, 1999] partial results about com-

plexity and tractable subclasses were given and a first ad hoc algorithm was provided, based on a discretization of time. This work has been recently completed by the introduction of the *Waypoint controllability* feature [Morris and Muscettola, 1999] (i.e. there are some time-points which can be assigned the same time of occurrence in all solutions).

This paper follows from these former works and is divided into two main parts. Section 2 provides a unified definition of the four controllability properties, extending the expressiveness by introducing the new concept of *BeginClb* constraint. Section 3 gives a new complete Dynamic controllability checking method based on a translation of the constraint network into a *timed automaton* that is shown to express precisely the original problem. We provide a *synthesis* algorithm inspired by the automatic control community, that behaves in a continuous game manner, and we prove that it solves our problem. The paper ends with a discussion about relevance of the approach and efficiency issues.

2 CONTINGENT CONSTRAINTS AND CONTROLLABILITY

We first recall the basics of Temporal Constraint Networks [Schwalb and Dechter, 1997]. At the qualitative level, we rely on the time-point continuous algebra [Vilain et al., 1989], where time-points are related by a number of relations, that can be represented through a graph where nodes are time-points and edges correspond to precedence ($\preceq$) relations. We can use the same time-point graph to represent quantitative constraints as well, thanks to the TCN formalism [Schwalb and Dechter, 1997]. Here continuous binary constraints define the possible durations between two time-points by means of temporal intervals. A basic constraint between x and y is $l_{xy} \leq (y - x) \leq u_{xy}$ equally expressed as $c_{xy} = [l_{xy}, u_{xy}]$ in the TCN. TCN a priori allow disjunctions of intervals, but we will restrict ourselves to the so-called *STP* (Simple Temporal Problem) where disjunctions are not permitted. A TCN is said to be consistent if one can CHOOSE for each time point a value such that all the constraints are satisfied, the resulting instantiation being a *solution* of the STP modelled by the TCN. Then consistency checking of such a restricted TCN can be made through complete and polynomial-time propagation algorithms (e.g. the path-consistency algorithm PC-2 [Mackworth and Freuder, 1985], used *incrementally* in our case, i.e. at each addition of a new constraint).

The TCN suits well the cases in which effective dates of time-points and effective durations of constraints are always chosen by the agent. If not, the problem has to be redefined in the following way.

2.1 A TAXONOMY OF TEMPORAL CONSTRAINTS

One needs first distinguishing between two different kinds of time-points: the *activated* ones are such that the agent can freely instantiate them; *received* time-points are those which effective time of occurrence is out of control and can only be observed.

This raises a corresponding distinction between so-called *controllable* and *contingent* constraints (*Clb* and *Ctg* for short): the former can be restricted or instantiated *by the agent* while values for the latter will be provided (within allowed bounds) *by the external world* (see [Vidal and Fargier, 1999] for details).

For instance, in planning, a task which duration is uncertain and will only be known when the task is completed at execution time will be modelled by a Ctg between the beginning time-point which is an activated one and the ending time-point which is a received one. As far as Clbs are concerned, we introduce here a more complete characterization, distinguishing between

- An *EndClb* between x and y is a Clb in the usual way we designed them until now: after x has occurred, the system can let time fly before assigning a value to the EndClb. If y is an activated time-point, then the EndClb is called a Free, and the agent can let time fly up to the upper bound of the constraint u_{xy} before deciding when to release y. If y is a received time-point, then it means the agent can freely restrict the duration interval (hence it is not a Ctg), but the effective value will be assigned only when y is received. This kind of EndClb is called a Wait. For instance, a delay between the end of a task and the beginning of the next one is usally a Free, while a constraint restricting the possible delay between the ends of two tasks (and which can be further restricted if needed) is a Wait[1].
- A *BeginClb* between x and y is a Clb such that the effective duration must be decided as soon as x has occurred. For instance, a task corresponding to a robot move may have different controllable durations depending on the robot speed that can be fixed by the agent, but this has to be done at the beginning of the task and will not be changed

[1] In the following, the distinction between the Free and the Wait is not needed, all EndClb will be considered in the same way.

throughout the move. Hence the duration is determined when releasing the task.

This defines a new taxonomy of temporal constraints, accounting for types of time-points, that appears in figure 1, where b stands for an activated time-point, e a received one, and x for any kind of time-point.

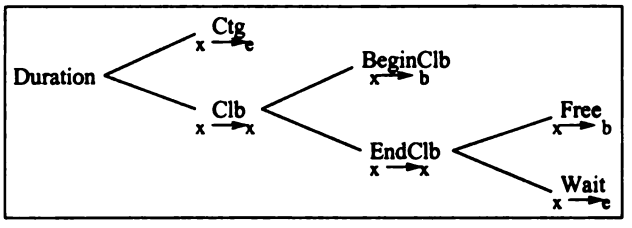


Figure 1: A taxonomy of temporal constraints

2.2 THE CONTINGENT TCN MODEL

The previous subsection introduces some level of uncertainty in the STP that results in the following definition of the corresponding *Contingent TCN*.

Definition 1 (CTCN)
$\mathcal{N} = (V_b, V_e, R_g, R_c)$ *is a Contingent Temporal Constraint Network with*

$V_b = \{b_1, \ldots, b_B\} \cup \{b_0\}$*: set of the* B *activated time-points plus the origin of time,*

$V_e = \{e_1, \ldots, e_E\}$*: set of the* E *received time-points,*

$R_c = \{c_1, \ldots, c_C\}$*: set of the* C *Clbs, with* $R_c = R_{bc} \cup R_{ec}$*,*

R_{bc} *being the set of EndClb*

and R_{bc} *being the set of BeginClb,*

$R_g = \{g_1, \ldots, g_G\}$*: set of the* G *Ctgs.*

In classical CSP frameworks, domains of the variables are also provided as unary constraints on the initial possible times of occurrence of time-points. Anyway they are equivalent to binary constraints between the origin of time b_0 and the variable (see [Meiri, 1991]).

In the following, a *decision* will refer to the effective time of occurrence of an activated time-point b_i and will be noted $\delta(b_i)$, and an *observation* will refer to the effective duration of the Ctg between x_i and e_i (known by the system when receiving time-point e_i) and will be noted ω_i.

2.3 A UNIFIED DEFINITION OF CONTROLLABILITY PROPERTIES

In a CTCN, the classical consistency property is of no use. It would mean indeed that there exists at least one complete assignment of times of occurrence to the whole set of time-points such that each effective duration belongs to the corresponding interval constraint. But this would require that in such a "solution" a value is chosen for the Ctgs, which contradicts the inherent unpredictable nature of those constraints. The decision variables of our problem are only the activated time-points. And hence a solution should be here, intuitively, an assignment of the mere activated time-points such that all the Clbs are satisfied, whatever values are taken by the Ctgs. This suggests the definition of the so-called *controllability* property.

In [Vidal and Fargier, 1999], three different levels of controllability have been exhibited, completed in [Morris and Muscettola, 1999] by an additional property called the Waypoint Controllability. Here we introduce a unified definition of those four properties, taking into account for the first time the notion of EndClb and BeginClb.

Definition 2 (Situations)
Given that $\forall i = 1 \ldots G$, $g_i = [l_i, u_i]$,

$\Omega = [l_1, u_1] \times \ldots \times [l_G, u_G]$ *is called the* space of situations, *and*

$\omega = \{\omega_1 \in [l_1, u_1], \ldots, \omega_G \in [l_G, u_G]\} \in \Omega$ *is called a* situation *of the CTCN.*

Then, for each time t*, one can define the* current-situation $\omega_{\prec t} \in \Omega_{\prec t}$ *which is the set of observations prior to* t*, i.e. such that only Ctgs with ending points* $e_i \preceq t$ *are considered,*

In other words, a *situation* represents one possible assignment of the whole set of Ctgs, and a *current-situation* with respect to t is a possible set of observations up to time t.

Definition 3 (Schedules)
$\Delta = \{\delta = \{\delta(b_1), \ldots, \delta(b_B)\}$ *is called the* space of schedules *and is the set of all the possible complete assignments of dates to activated time-points* δ *(i.e. the cartesian product of all the interval constraints between* b_0 *and* b_i*).*

$\delta \in \Delta$ *is called a* schedule.

Then, for each time t*, one can define the* current-schedule *which is the subsequence assigned so far* $\delta_{\preceq t}$ *s.t.* $\forall x_i \in V_b \cup V_e$ *with* $x_i \preceq t$,

if $\exists c_{ij} = (b_j - x_i) \in R_{bc}$ *then* $\delta(b_j) \in \delta_{\preceq t}$

else if $x_i \in V_b$ *then* $\delta(x_i) \in \delta_{\preceq t}$

A *schedule* is then one possible sequence of decisions (that might be "controllable" or not). And a *current-schedule* encompasses the notion of reactive chronological building of a solution in dynamic domains. One should notice that we take into account the case of BeginClb, for which the time of occurrence of the ending

point b_j might have been already decided at time t (and hence must be in $\delta_{\prec t}$) even if $t \preceq b_j$.

Definition 4 (Projection and Mapping)
$\mathcal{N}_\omega$ is the projection *of $\mathcal{N}$ in the situation ω, built by replacing each Ctg g_i by the corresponding value $\{\omega_i\} \in \omega$. In [Vidal and Fargier, 1999] a projection is proved to be a simple TCN corresponding to a STP.*
μ is a mapping from Ω to Δ such that $\mu(\omega) = \delta$ is a schedule applied in situation ω.

Intuitively, a CTCN will be "controllable" if and only if there exists a mapping μ such that every schedule $\mu(\omega)$ is a solution of the projection $\mathcal{N}_\omega$. In fact this is only the "weakest" view of the problem, in many applications one needs more restricted properties, namely the Dynamic and the Strong controllabilities. Another interesting property called Waypoint controllability has been further introduced in [Morris and Muscettola, 1999]. The following definition is a unified view of those four properties.

Definition 5 (Controllabilities)

- $\mathcal{N}$ *is* Weakly controllable *iff*
  $\exists \mu$ *s.t.* $\forall \omega \in \Omega, \mu(\omega) = \delta$ *is a solution of* $\mathcal{N}_\omega$.
- $\mathcal{N}$ *is* Strongly controllable *iff*
  (1) $\mathcal{N}$ *is Weakly controllable*
  (2) μ *is a constant mapping:*
  $\forall \omega \in \Omega$, $\mu(\omega) = \delta$ *is unique*
- $\mathcal{N}$ *is* Dynamically controllable *iff*
  (1) $\mathcal{N}$ *is Weakly controllable*
  (2) $\forall (\omega, \omega') \in \Omega^2$, *with* $\delta = \mu(\omega)$ *and* $\delta' = \mu(\omega')$, $\forall t$,
  if $\exists \omega_{\prec t}$ *s.t.* $\omega_{\prec t} \subset \omega$ *and* $\omega_{\prec t} \subset \omega'$
  then $\delta_{\prec t} = \delta'_{\prec t}$
- $\mathcal{N}$ *is* Waypoint controllable *iff*
  (1) $\mathcal{N}$ *is Weakly controllable*
  (2) $\exists W \subset V_b$ *s.t.*
  $\forall (\omega, \omega') \in \Omega^2$, *with* $\delta = \mu(\omega)$ *and* $\delta' = \mu(\omega')$,
  $\forall x \in W$, $\delta(x) = \delta'(x)$

In other words, $\mathcal{N}$ is Stongly controllable iff there exists one "universal" solution that will fit any situation. It is Weakly controllable iff, knowing the complete situation, one can pick up a schedule that fits this situation. As discussed in [Vidal and Fargier, 1999], in a plan execution for instance, one will observe the situation in the process of time, and hence if a schedule decision depends on some part of the situation that is still to come and hence unknown, one will not be able to pick up the "right" decision. The Dynamic controllability definition above solves the problem, compelling a current schedule at any time t to depend only upon the current situation at that time. More details and checking algorithms for those three properties can be found in [Vidal and Fargier, 1999].

Waypoint controllability states that there are some points for which all the schedules share the same time of occurrence, whatever the situation is[2]. Those waypoints serve as "meeting" time-points in a plan, when the agent waits for all the components of a subpart to be over before starting the next stage. Waypoints are created during the planning process through the addition of "wait periods". In [Morris and Muscettola, 1999] an algorithm for checking Waypoint controllability is given, which is shown to be exponential.

One of the main advantages of the unified and progressive definition above is that the following property comes now directly from it (the proof is straightforward considering that acceptable mappings for the Strong controllability are restrictions of acceptable mappings for both the Waypoint and the Dynamic controllabilities, which in turn are restrictions of acceptable mappings for the Weak controllability).

Property 1 (Implication rules)
Strong controllability $\Rightarrow$ *Dynamic controllability* $\Rightarrow$ *Weak controllability*
Strong controllability $\Rightarrow$ *Waypoint Controllability* $\Rightarrow$ *Weak controllability*

We will now focus on Dynamic controllability issues. The "home made" checking algorithm given in [Vidal and Fargier, 1999] relies on a discretization of time. Considering the inherent continuous nature of interval constraints in TCNs, we have been interested in the applicability of timed automata in this context, which will be developped in the next section.

But before that, let us study an example to illustrate the controllability properties. Figure 2 exhibits three small CTCNs. In the first one (a), two tasks with contingent durations are constrained with each-other by a *during*-like [Allen, 1983] relation[3]. For instance that might be a data sending task to an orbiter for a planetary robot that needs to be achieved during the orbiter visibility temporal window, both durations being uncertain. The second example (b) shows two successive contingent tasks, with a maximum (controllable) delay of 10 time units between them. For instance a robot might have to leave a room within 10 seconds after

[2]We have chosen to restrict in some sense the original Waypoint controllability definition that allowed the set W to contain received events, since in general those would not satisfy the Waypoint controllability property.

[3]Unlabelled arrows stand for simple precedence, that should simply be $[0, +\infty[$ intervals in the TCN framework.

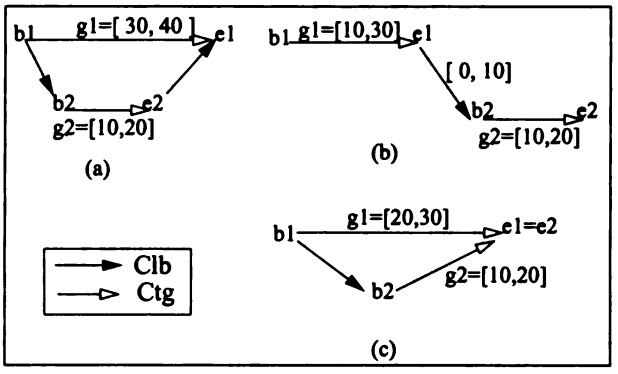


Figure 2: Illustrating controllability

having set a security alarm inside. The third example (c) shows two tasks, one activated after the other, that should finish exactly at the same time, though they both have uncertain durations. Obviously this last example cannot be accepted as a valid plan, since no execution can guarantee that the constraint will be met. Interestingly enough, the three examples are all controllable in the weak sense (i.e. condition 1 holds): for any values of the Ctgs, one can find a set of decisions that is consistent. But only examples (a) and (b) meet condition 2 of Dynamic controllability, which fits what one should expect in realistic planning. Moreover, in example (a), one should get that b_2 must be activated at most 10 time units after b_1 to ensure the satisfaction of the relation whatever values are taken by ω (since in the worst case $\omega_1 = 30$ and $\omega_2 = 20$). This result would not be issued by a classical TCN consistency propagation algorithm, that would return a maximum value of 30. Last, about Waypoint controllability, in example (a) b_1 and b_2 can be waypoints, since it is possible (and actually needed) to fix their times of occurrence independently of the situations that may be observed. This is not the case in (b), since the time at which one can release the second task depends upon the effective duration of the first one, and hence cannot be set in advance, which means only $W = \{b_1\}$ can be defined as a waypoint set. In example (c), setting $W = \{b_1\}$ satisfies as well Waypoint controllability: actually it lets b_2 depend on the outcome of ω_1 and ω_2. Which means Waypoint controllability alone is not satisfactory in practice, since it does not take into account the chronological order of events. Anyway, in [Morris and Muscettola, 1999], it is proven to be equivalent to Dynamic controllability under some restrictive conditions. Unfortunately these conditions do not hold in case (c), hence Dynamic controllability still has to be checked.

3 THE TIMED GAME AUTOMATON METHOD

3.1 THE PRINCIPLES OF TIMED AUTOMATA

We will assume in the following the reader has a minimal knowledge of finite-state automata. The method we present here relies on the *timed automata* model used for describing the dynamical behaviour of a system [Alur and Dill, 1994]. It consists in equiping a finite-state automaton with time, allowing to consider cases in which the system can remain in a state during a time T before making the next transition. This is made possible by augmenting states and transitions with "continuous variables called *clocks* which grow uniformly when the automaton is in some state. The clocks interact with the transitions by participating in pre-conditions (*guards*) for certain transitions and they are possibly reset when some transitions are taken." This is illustrated by the following example.

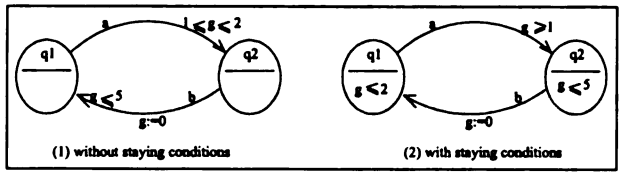


Figure 3: A basic example of timed automaton

The system is initially in state q_1 and clock g is set to 0. It has to stay in q_1 at least 1 time unit and at most 2, then it can move to q_2. It should leave it before $g = 5$, which resets the clock and the system is back in the initial state. Letters a and b are classical labels "read" by the system when the transition is taken. The original model can be slightly changed converting some guards into *staying conditions* [Asarin et al., 1995] in states. The expressiveness remains the same, but those are more convenient to use when addressing controllability checking (see section 3.4).

In our model (see next subsection), we will introduce a second type of clock that work in the opposite way: it is first set to a fixed value, then it decreases uniformly until it reaches the value 0, which serves as a specific guard for activating a transition. To distinguish them, we have chosen to call the first type of clock *stopwatch* and the second one *egg timer* ... The former will allow to model Ctgs and EndClbs, while the latter will be necessary to model BeginClbs.

In [Maler et al., 1995, Asarin et al., 1995], such tools are claimed to fit *real-time games*, where transitions are divided in two groups (such as constraints in

CTCN) depending on which of the two players control it, and some states are designated as *winning* for one of the players. The strategy for each player is to select controlled transitions that will lead her to one of her winning states. This extension of the classical discrete game approach has the following features: (1) there are no "turns" and the adversary need not wait for the player's next move [Maler et al., 1995], and (2) each player not only choose between alternative transitions, but also between *waiting* or not before taking it, from which one can view the two-player game as now "a three-player one where Time can interfere in favor of both of the two players [Asarin et al., 1995]".

This is especially interesting for controlling reactive systems in which one player is the *environment* ("Nature") and the other player is the *controller* (for instance a plan execution controller) and has control only over some of the transitions (similarly as Clbs in CTCN). A *trajectory* (i.e. a path in the automaton) reaching a winning state for the controller is called *C-trajectory* in [Maler et al., 1995], and defines a so-called *safety game* [Asarin et al., 1995] policy.

3.2 AN ACCURATE TIMED GAME AUTOMATON MODEL

Different formulations of timed automata models can be found in the bibliography of the area. We propose here our own model for which we chose to represent what is needed and only what is needed for our purpose. We have chosen to cut the model definition in two steps: first we define the types of clocks and the conditions and functions that can be used on them, then we give the definition of the Timed Automaton.

Definition 6 (Clock conditions and functions) *$\Gamma = \Gamma_{sw} \cup \Gamma_{et}$ is the discrete and finite set of clocks and may be of any cardinality (i.e. one can define as many clocks as one needs), where Γ_{sw} is the set of* stopwatches *and Γ_{et} is the set of* egg timers, *on which are defined three sets of conditions and actions that can be associated to clocks:*

- $\mathtt{Re} = \{(sw_i \leftarrow 0)$ *s.t.* $sw_i \in \Gamma_{sw}\}$ *is the finite set of all possible stopwatch reset functions,*
- $\mathtt{As} = \{(et_i \leftarrow \Delta_i)$ *s.t.* $et_i \in \Gamma_{et}$ *and* $\Delta_i \in \mathbb{Z}\}$ *is the infinite set of all possible egg timer assignment functions,*
- Cond *defines the set of* k*-polyhedral [Maler et al., 1995, Asarin et al., 1995] relation sets on clocks and is the infinite set of all possible clock conditions that will be used as guards or staying conditions.*

The so-called k-polyhedral sets define a restriction of the possible ways to express guards and staying conditions (see [Maler et al., 1995, Asarin et al., 1995] for details): it corresponds to constraining sums and/or differences of clocks to be bounded by rationals[4]. In our case, we will barely need conditions such as:

- $\{(l_i \leq clock_i \leq u_i)$ s.t. $clock_i \in \Gamma$ and $(l_i, u_i) \in \mathbb{Z}^2\}$ $\cup \{(et_i = 0)$ s.t. $et_i \in \Gamma_{et}\}$ are conditions that will appear as guards, expressing either ranges of values that must/will be reached by a stopwatch, or that an egg timer has reached the value 0.
- $\{(L \leq clock_i \pm clock_j \leq U)$ s.t. $clock_{i/j} \in \Gamma$ and $(L, U) \in \mathbb{Z}^2\}$ will be the conditions that will appear as staying conditions in states.

Definition 7 (Timed Game Automaton) *$\mathcal{A} = (\mathcal{Q}, \Sigma, \Gamma, \mathrm{S}, \mathrm{T})$ is a timed game automaton (TGA) where*

- *$\mathcal{Q}$ is the discrete and finite set of states q_i, with three special cases:*
  - *q_0 is the initial state,*
  - *q_{ok} is the unique* winning *state,*
  - *$q_\dagger$ is the unique* losing *state;*
- *$\Sigma = \Sigma_b \times \Sigma_e$ is the input alphabet such that any label in Σ_b is of the form b_i and any label in Σ_e is of the form e_i;*
- *Γ is defined as above;*
- *$\mathrm{S} : \mathcal{Q} \rightarrow$ Cond assigns staying conditions to states;*
- *$\mathrm{T} = \mathrm{T}_b \cup \mathrm{T}_e \subseteq \mathcal{Q}^2 \times \Sigma \times$ Cond $\times$ Re $\times$ As is the set of transitions of the form $\tau = <q, q', \sigma, g, r, a>$ with a distinction between*
  - *$\tau \in \mathrm{T}_b$ is an* activated *transition* iff *$\sigma \in \Sigma_b$,*
  - *$\tau \in \mathrm{T}_e$ is a* received *transition* iff *$\sigma \in \Sigma_e$.*

Therefore for activated transitions, if there is a guard conditionning its activation, the agent will be able to decide the exact time of activation by "striking" the stopwatch within the two bounds of the guard. If it is a received transitions, then the transition will be automatically taken at some unpredictable time within the two bounds of the guard. Egg timers will on the contrary compel some activated transitions to be taken at one and only one predicted time.

Considering the losing state, obviously an agent will never make a move leading to it, and no clocks need to be reset or assigned at that point, which means that

$$\text{if } \tau = <q, q_\dagger, \sigma, g, r, a>, \text{ then } \tau \in \mathrm{T}_e,\ r = a = \emptyset$$

[4]This restriction compels to correspondingly accept only rationals as bounds of duration intervals in the CTCN, which is easily met in practical applications.

3.3 DESIGNING AN EQUIVALENT AUTOMATON FROM THE CTCN

Going back to our Dynamic controllability problem, we can compare the CTCN formalism with the TGA by stating that the former describes the *specifications* of a dynamic system (for instance a plan that must be run), through the constraints it has to meet, whereas the latter captures all the possible *execution scenarii* of this system: one can view it as a *simulation* model.

We will prove in the next subsection that Dynamic controllability can be easily checked in the TGA, but first we need to show that the CTCN can be translated into a TGA, which means we need to prove the following property.

Property 2 (Expressiveness implication) *For any CTCN $\mathcal{N}$, there exists a TGA $\mathcal{A}$ expressing the simulation of $\mathcal{N}$.*

Proof. First let us exhibit some intuitive correspondences between the two models. Obviously letters b and e and terms *activated* and *received* refer to the same concept in both models: the events in $\mathcal{N}$ will appear as translations labelled with this event in $\mathcal{A}$. Therefore $\Sigma_b \equiv V_b$ and $\Sigma_e \equiv V_e$. Similarly, temporal intervals in $\mathcal{N}$ will appear as guards in $\mathcal{A}$.

So one needs to determine how items in $V_b, V_e, R_g, R_{bc}, R_{ec}$ can be expressed in the automaton model: we chose to provide a constructive proof that will suggest at the same time the automaton building algorithm.

Let us first consider *Untime*($\mathcal{A}$) the TGA $\mathcal{A}$ from which clocks and guards are discarded. The *formal language* interpretation of *Untime*($\mathcal{A}$) (see [Alur and Dill, 1994] for details) defines *words* that are accepted: a word is such that each letter represents a transition, and the sequence of letters corresponds to a trajectory from q_0 to q_{ok}. Similarly, in a CTCN considered at the symbolic level (i.e. only time-points and precedence constraints), one can look for all the possible sequences of time-points that satisfy the symbolic constraints (this might be sped up by extracting a so-called *dispatchable* network [Morris et al., 1998] from $\mathcal{N}$). For that, one can start from b_0 and choose one time-point among its *successors* (succ(i)=$\{j \mid i \preceq j \wedge not(\exists k : i \preceq k \preceq j)\}$) and reapply this process recursively until reaching some ending point (a time-point with no successor): a totally ordered sequence of such events is then an accepted word in the above "untimed" meaning.

To build *Untime*($\mathcal{A}$) from $\mathcal{N}$, we thus need to create a new transition (which type, activated or received, corresponds to the event type) and a new state at each step of the recursive process. Anyway, constraint violation in $\mathcal{N}$ must be accounted for in *Untime*($\mathcal{A}$) as well, in order to represent the trajectories leading to the losing state. We have argued that only received transitions, hence labelled with e events, may lead to $q_\dagger$. So for each Ctg or Wait $(e_j - x_i)$, for each time-point y successor or equal to x_i and predecessor of e_j, one must consider e_j as a possible next letter as well. And if e_j is not a successor of y (i.e. $\exists k : y \preceq k \preceq e_j$)), then a constraint is violated and the corresponding transition leads to $q_\dagger$, and a recursive building process will have to backtrack here.

Then, to get $\mathcal{A}$ from *Untime*($\mathcal{A}$), one needs to translate quantitative constraints in $\mathcal{N}$ into distance conditions between the letters of those words. Let us consider the different types of constraints $(x_j - x_i) \in [l_i, u_i]$:

Ctg / EndClb To each transition labelled x_i, one just has to add a reset function $(sw_i \leftarrow 0) \in$ Re, and to each transition labelled x_j, a guard $(l_i \leq sw_i \leq u_i) \in$ Cond: hence the transition x_j is compelled to occur between l_i and u_i time units after x_i, however distant they are from one another in the trajectory, and in the EndClb case, one can wait the very last transition reaching x_j before deciding the exact delay.

BeginClb Here it is slightly different since the delay between x_i and x_j must be determined as soon as x_i has occurred, which is modelled by assigning a value to an egg timer $(et_i \leftarrow \Delta_i) \in$ As on each transition labelled x_i together with a guard $(l_i \leq et_i \leq u_i) \in$ Cond that will restrict the possible values Δ_i, and then adding a guard $(et_i = 0) \in$ Cond to each transition labelled x_j, which gives the expected effect.

To complete the proof, one must notice that since CTCNs implicitely allow the simulataneous occurrence of two events (if the delay between them is set to 0), so must do the automaton. This simultaneity is argued in [Alur and Dill, 1994] to be easily permitted in timed automata, thanks to the separation between the "untimed" and "timed" language interpretations (for instance to the word $(b_1 b_2 e_2 e_1)$ can be associated the same time values for b_1 and b_2). In game automata this expressiveness is explicitely mentioned [Asarin et al., 1995]. ◇

This constructive process not only gives evidence that TGA can express exactly what a CTCN expresses (actually more), but it is now easy to provide an algorithm based on that process that build $\mathcal{A}$ from $\mathcal{N}$.

Build($\mathcal{N}$,$\mathcal{A}$)

$Q \leftarrow \{q_0, q_{ok}, q_\dagger\}$
$Exp \leftarrow \emptyset$
$Act \leftarrow \{i_0\}$
$s \leftarrow 0$
develop(q_0, i_0, Act, Exp)

where i_0 is the initial time-point of $\mathcal{N}$, *Act* [resp. *Exp*] is the set of currently "activable" time-points in V_b [resp. currently expected time-points in V_e], and the recursive function **develop** is as follows:

Develop(q, i, Act, Exp)

```
forall i' ∈ next(i)
   if i'∈ V_b then add(i', Act) else add(i', Exp)
remove(i, Act ∪ Exp)
forall j ∈ Act [resp. Exp]
   | compute_transition(j, g, r, a)
   | if j = i_∞ then add(<q, q_ok, g, ∅, ∅>, T_e)
   |    else if violated(j) then
   |                        add(<q, q_↑, g, ∅, ∅>, T_e)
   |       else
   |       | s ← s + 1
   |       | add(q_s, Q)
   |       | add(<q, q_s, g, r, a>, T_b [resp. T_e])
   |       | DEVELOP(q_s, j, Act, Exp)
```

The function *next*(i) returns the direct children of i in $\mathcal{N}$ [Ghallab and Vidal, 1995], i_∞ is the final time-point in $\mathcal{N}$, functions *add* and *remove* are as expected, and **violated**(j) is true if at least one point still in *Exp* or *Act* is constrained to be *after* j in $\mathcal{N}$.

Then the function **compute_transition**(g, j, q) computes the temporal part of the new transition: the guard and the reset and assign functions. This is related to the kind of time-point j, based upon the cases appearing in the property proof above, which is summarized hereafter:

- If j is the beginning of a BeginClb, then an egg timer is set through an assign function, together with a guard reflecting the bounds of the BeginClb.
- If j is the end of a BeginClb, then a guard is added corresponding to the egg timer reaching the value 0.
- If j is the beginning of a Ctg or an EndClb, then a new stopwatch is initialized through a reset function.
- If j is the beginning of a Ctg or an EndClb, then a guard requiring that the corresponding stopwatch value lies within the constraint bounds is added.

This algorithm can only compute a TGA "from scratch" from a given CTCN. In practical applications though, constraints are added one by one and controllability should be checked at each step. This calls for an *incremental* version, completing the current TGA instead of recomputing it each time a new constraint is added, which is the topic of on-going work.

3.4 AUTOMATON SYNTHESIS PROVES DYNAMIC CONTROLLABILITY

We can now exhibit an algorithm acting on the TGA, and prove that this algorithm checks Dynamic controllability. This method is inspired by well-established techniques in the area of reactive program synthesis. A *synthesis* algorithm uses as a basic principle a so-called *controllable predecessors* operator: unformally, this operator computes from a state q the states from where the system can be compelled to reach q, "returning revised guards and staying conditions [Asarin et al., 1995]". The algorithm will recursively apply this operator from the initial set of winning states until it reaches a fixed point (which is always met, which means the algorithm is decidable [Maler et al., 1995]). If q_0 is in the final set, then the controller can always win the game.

We have designed our own synthesis algorithm that best fits our TGA model. Calling the recursive function **Pred_Op**(q_{ok}) adds in each state staying conditions that are sufficient to avoid transitions to the losing state.

Pred_Op(q)

```
if q ≡ q_0 then return("Synthesis completed :-)")
forall q⁻ s.t.∃τ =<q⁻, q, σ, g, r, a>
   | propagate_back(S(q), S(q⁻))
   | if ∃τ' =<q⁻, q_↑, σ', g', r', a'> then
   |    | S(q⁻) ← S(q⁻)∩ avoid-losing(g, g')
   |    | remove τ' from T_e
   | if inconsistent(S(q⁻)) then
   |    | if q⁻ ≡ q_0 then return("Fail :-(")
   |    | q⁻ ⇐ q_↑
   |    | remove τ from T
   |    else Pred_Op(q⁻)
```

The two functions **propagate_back** and **avoid-losing** and the test **inconsistent** are used in the following ways:

- **propagate_back**$(\mathrm{S}(q), \mathrm{S}(q^-))$ computes the staying condition on q^- from the one on q and the guard g, the reset function r and the assign function a on the transition τ, according to the type of τ.
- **avoid-losing**(g, g') considers the two guards on the two transitions, and according to the clocks involved and their types (stopwatch or egg timer) computes the staying

condition that must be satisfied on q^- to ensure that τ' cannot be taken. Hence τ' can be removed afterwards.

inconsistent($S(q^-)$) checks if the resulting set of staying conditions is feasible. If not, then the state is made equivalent to the losing state, and if it is the initial one, then the synthesis fails.

Let us get into more details. The most tricky part is the **avoid-losing** function. Up to now we have only completely characterized the cases in which merely stopwatches are involved (i.e. no BeginClb appear in the CTCN)[5]. Hence the guards g and g' will look like $(l \leq sw \leq u)$ and $(l' \leq sw' \leq u')$. Since the two clocks grow uniformly, the quantity $sw' - sw$ is constant and will support the condition. Since transition τ' must be avoided and is a received transition, one must keep $(sw' \leq l')$ in q^-. This means that when sw' takes the value l', one must be sure that τ has already been taken. Then it depends on the type of τ: if it is an activated transition, then one can activate it as soon as $(sw \geq l)$, while if it is a received transition, one cannot ensure it to be taken before $(sw \geq u)$. This raises the following staying condition on q^-:

$$\texttt{If } \tau \in T_b \texttt{ then } add((sw' - sw \leq l' - l), S(q^-)) \\ \texttt{else } add((sw' - sw \leq l' - u), S(q^-))$$

Then **propagate_back** and **inconsistent** are straightforward.

Since a transition is instantaneous, a staying condition that must be true when entering a state must necessarily be true when leaving the previous state. Hence in **propagate_back** it is enough to copy the staying condition in the previous states. The only specific case is when the transition is characterized by a reset function on one of the clocks involved in the staying condition: if $(sw \leftarrow 0)$ appears on the transition, then sw in the staying condition of the preceding state can be replaced by 0, and one gets for instance $(sw' \leq l' - u)$.

Last, the test **inconsistent** is simply true when after intersection and back propagation a staying condition becomes $(sw \leq m)$ with $m < 0$.

All this process will be illustrated in a very clear way in the next subsection trough our example.

As already said, *incrementallity* is often a key issue. If the automaton $\mathcal{A}$ has already been successfully synthesized, and one adds a set of new transitions and states, then the incremental version of our algorithm is straightforward: it merely applies the `Pred_Op` operator to the new set of states.

Now it just remains to show the next main property.

Property 3 (Dynamic controllability checking)

Dynamic controllability in $\mathcal{N}$ is solved by the synthesis algorithm in the equivalent automaton $\mathcal{A}$

Proof. We can first prove that the condition (2) of Dynamic controllability (see definition 5) is met in $\mathcal{A}$, which simply comes from the automaton structure, hence this first part of the proof does not involve the synthesis algorithm, we barely consider $\mathcal{A}$ as it is just after being built from $\mathcal{N}$. When simulating a trajectory in $\mathcal{A}$, one actually maps to some observed ω a δ (that might or not be a solution of $\mathcal{N}_\omega$). For each state q, for each time t such that at time t the automaton is in q, then all $e_i \preceq t$, all $b_i \preceq t$ such that b_i is the ending point of an EndClb, and all $x_i \preceq t$ such that x_i is the beginning point of a BeginClb, necessary label transitions already taken, and in the case of BeginClb the effective duration has already been fixed through the assign function on the transition. Which means, following definitions 2 and 3, that the couple $(\omega_{\prec t}, \delta_{\prec t})$ is set. Hence the $\delta_{\prec t}$ reached at time t only depends on $\omega_{\prec t}$, which entails condition (2). Briefly speaking, mappings provided by the automaton correspond exactly to those entailed by condition (2).

The second part of the proof corresponds to the condition (1) of Dynamic controllability: $\exists \mu : \Omega \longrightarrow \Delta$ s.t. $\forall \omega \in \Omega, \mu(\omega) = \delta$ is a solution of $\mathcal{N}_\omega$. If $\mathcal{A}$ is successfully synthesized, then all trajectories get to the winning state, which means all δ satisfy the constraints, and hence are solutions of the corresponding $\mathcal{N}_\omega$. On the opposite way, if the synthesis of $\mathcal{A}$ fails, then that means that there exists a state q with a transition to $q_\dagger$ labelled by some received event e that cannot be prevented to be taken, which means there is a ω_i corresponding to a possible time of occurrence for e that will push the automaton into that transition, and a constraint will be violated. This means that in q, whatever couple $(\omega_{\prec t}, \delta_{\prec t})$ has been set, $\omega_{\prec t} \cup \{\omega_i\}$ necessary violates a constraint, and therefore all complete situations ω such that $\omega_{\prec t} \cup \{\omega_i\} \subset \omega$ cannot be mapped to a schedule that is a solution to $\mathcal{N}_\omega$. $\diamond$

3.5 ILLUSTRATION THROUGH THE EXAMPLE

To get an idea of how the algorithms work in practice, one can reconsider the example of Figure 2(a). Representing it in a TGA formalism gives the first drawing of Figure 4, where on each transition one can view the guard condition and the label above

[5] Taking into account both stopwatches and egg timers is a little more complex and is still the subject of on-going work.

and the clock reset below. Two stopwatches are used that are called g_1 and g_2 by analogy with the corresponding contingent durations. No egg timer is needed since there are no BeginClb in the CTCN. One can notice that the system is not a priori prevented to receive e_1 before e_2 (which would violate a constraint), which is also the case in the original CTCN. Considering the *formal language* interpretation [Alur and Dill, 1994] of the TGA, that means that the TGA here a priori accepts the so-called untimed language (where words are succesions of labels irrespective of their precise times of occurrences) $\{(b_1e_1),(b_1b_2e_1),(b_1b_2e_2e_1)\}$. Then, "violating" simulation outcomes like (b_1e_1) and $(b_1b_2e_1)$ should be distinguished from correct ones like $(b_1b_2e_2e_1)$. Hence, receiving e_1 before e_2 appears as a transition to the losing state in the automaton.

Then the synthesis operator is applied from the winning state, adding new staying conditions each time there is a need to protect a state from a transition to $q_\dagger$. This is the case in q_2, where $(g_1 \leq 30)$ must be met, and considering that the uncontrolled clock g_2 might get up to 20 in q_2, one gets the staying condition $(g_2 - g_1 \leq 30-20=10)$ which is added in q_2 (and hence the transition to $q_\dagger$ can be removed) and propagated backward to q_1: here it becomes $(g_1 \leq 10)$ since g_2 is set to 0 on the transition between q_1 and q_2. This condition corresponds to a restriction of the Clb $(b_2 - b_1)$ in the original CTCN. In q_1 $(g_1 \leq 30)$ must also be met to avoid getting into the losing state. Since there is no guard on the transition between q_1 and q_2, this condition is simply intersected with $(g_1 \leq 10)$ which is left unchanged. Propagating it back to q_0 produces an empty staying condition because of the resetting function on g_1. The algorithm succeeds since one reaches q_0 without detecting any inconsistency and all transitions to $q_\dagger$ have been removed.

Modifying the example by replacing the second Ctg by $g_2 = [25, 35]$ would as one should expect it lead the synthesis algorithm to fail, since one would get $(g_2 - g_1 \leq 30-35=-5)$ in q_2 and therefore $(g_1 \leq -5)$ in q_1. Hence one would get as a result that the CTCN is not Dynamically controllable.

3.6 DISCUSSION ON PRACTICAL USE AND EFFICIENCY ISSUES

The model and the algorithms, however sound they may be, will only be relevant if one can give evidence of their usefulness in practice in dynamic applications, which is mainly a question of efficiency, knowing that the complexity of Dynamic controllability in the general case is untractable [Morris and Muscettola, 1999].

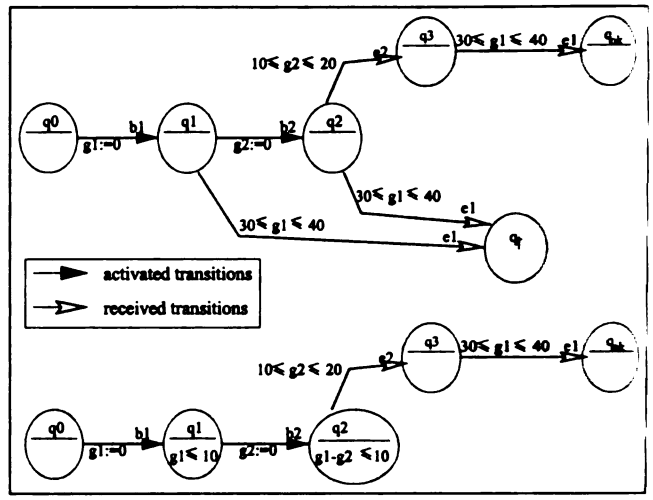


Figure 4: TGA interpretation of a CTCN

Obviously TCN is a more compact representation than automata: if we define the *degree of parallelism* p of $\mathcal{N}$ as being the maximum number of time-points expected and possibly activated / received at the same time (i.e. $p = max\{card(concur(i))/i \in V_b \cup V_e\}$, where concur(i)={j / $not(j \prec i) \wedge not(i \prec j)$}), and remembering that B is the number of decision variables, then the number of states of $\mathcal{A}$ is in the worst case p^B.

If one now considers the algorithms presented in this paper (here the versions "from scratch"), obvioulsy the automaton building algorithm has time and space complexities that are in the order of the number of states created, i.e. $O(p^B)$. As far as the synthesis algorithm is concerned, the complexity should be higher since one state may be traversed several times. Actually if the algorithm is designed in a breadth-first manner (i.e. propagate back from q only when all staying conditions have been computed in q), each state will be visited in the worst case p times, which corresponds here to the maximum number of transitions outgoing from a state. Which means the complexity is in $O(p.p^B) = O(p^B)$ here again. Which means the algorithms are exponential in the general case with respect to the CTCN data.

First, the method might be relevant when p is kept rather low, which will be the case in application domains where tasks carried out by a single agent are considered, for instance in a robot or space vehicle task planning application [Morris et al., 1998]: the network will mainly consists of a sequence of tasks with reduced concurrency. In multimedia authoring tools [Jourdan et al., 1997], the objects (audio, video, text) in a document might show a significant level of concurrency, but a document is mainly a sequence of "pages",

actually subparts of the document synchronizing at some points, which will allow to partition the problem. This might be less relevant in manufacturing process scheduling applications since one usually has to consider distinct sequences processed in parallel, but still sequentiality remains the key word, and moreover if the sequences in parallel are unrelated, then it might be possible to process them separately.

From another standpoint, automata-based techniques might be improved to reduce the number of states produced, considering that two subsequences containing the same set of events, though in distinct orders, might converge on the same state, or using more complex abstraction views, like in the so-called *symbolic approaches* (like *Decision Binary Diagrams*) [Maler et al., 1995, Asarin et al., 1995].

Another possibility is to accept an incomplete checking algorithm in the long term, using the TGA only in the short term, as far as execution runs, so as to account for safety: the algorithm anticipates the possible losing state deadends and can activate any necessary recovery action in advance.

Last, [Morris and Muscettola, 1999] argue that choosing cleverly the set of waypoints through addition of some "wait" periods in the plan might lead to Dynamic controllability being equivalent to Waypoint controllability. Anyway, trying to design a plan in this way might lead to a high number of waypoints lowering the plan optimality, and then Waypoint controllability would remain a rather hard problem. Another possibility would be to use waypoints only to restrict Dynamic controllability checking in all subparts of the networks between any pair of waypoints. This would lead to partition the problem and solve a number of small size networks: actually the quantity B would be bounded and kept low, which would raise a polynomial complexity in practice. This idea, that could be directly applied to the multimedia domain, and easily to others like space missions, is certainly the most promising one and is developped in more details in [Vidal, 2000].

4 CONCLUSION

This paper has given a thorough and complete characterization of controllability properties in CTCNs, introducing the new concept of BeginClb, and giving a unified definition that will ease further works on that field. We then have proven that Timed Game Automata had the necessary expressive power and that Dynamic controllability could be checked in such a model adapted to our temporal constraint framework, thanks to a so-called synthesis algorithm.

Anyway, our adapted synthesis algorithm has been completely characterized only in the easiest case where one need not deal with BeginClb constraints. It remains to thoroughly develop the cases in which egg timers must be accounted for. Another on-going work deals with providing an incremental version of the automaton building process. But the main next steps will be the design of a more efficient global framework using waypoints (see [Vidal, 2000]) and the development and testing of a prototype that might be compared to programs specialized in controller synthesis such as KRONOS [Yovine, 1997].

As it is suggested in the previous subsection, this approach will be mainly relevant in dynamic domains, with uncertainty on continuous temporal constraints, such as planning [Morris and Muscettola, 1999], scheduling, or multimedia authoring systems [Jourdan et al., 1997, Fargier et al., 1998]. Moreover, the automaton model is a natural tool to be further used on-line as a real-time execution controller of the plan [Morris et al., 1998] or multimedia document browsing for instance.

We would like to conclude mentionning the cross-disciplinary aspect of our method, contributing to bridge the gap between such distinct communities as artificial intelligence and theoretical computer science.

Acknowledgement

The author is grateful to Paul Morris (NASA Ames Research Center) for fruitful discussions and his suggestion on the new Dynamic controllability definition.

References

[Allen, 1983] Allen, J. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):509–521.

[Alur and Dill, 1994] Alur, R. and Dill, D. (1994). A theory of timed automata. *Theoretical Computer Science*, 126:183–235.

[Asarin et al., 1995] Asarin, E., Maler, O., and Pnueli, A. (1995). Symbolic controller synthesis for discrete and timed systems. In Antsaklis, P., Kohn, W., Nerode, A., and Sastry, S., editors, *Hybrid Systems II, LNCS 999*. Springer Verlag.

[Brusoni et al., 1994] Brusoni, V., Console, L., Pernici, B., and Terenziani, P. (1994). LaTeR: a general purpose manager of temporal information. In *Methodologies for Intelligent Systems 8, Lecture*

Notes in Computer Science, volume 869, pages 255–264. Springer Verlag.

[Dousson et al., 1993] Dousson, C., Gaborit, P., and Ghallab, M. (1993). Situation recognition: representation and algorithms. In *Proceedings of the 13th International Joint Conference on A.I. (IJCAI-93)*, Chambéry (France).

[Dubois et al., 1993] Dubois, D., Fargier, H., and Prade, H. (1993). The use of fuzzy constraints in job-shop scheduling. In *IJCAI-93 Workshop on Knowledge-Based Planning, Scheduling and Control*, Chambéry (France).

[Fargier et al., 1998] Fargier, H., Jourdan, M., Layaïda, N., and Vidal, T. (1998). Using temporal constraint networks to manage temporal scenario of multimedia documents. In *ECAI-98 workshop on Spatio-Temporal Reasoning*, Brighton (UK).

[Fargier et al., 1996] Fargier, H., Lang, J., and Schiex, T. (1996). Mixed constraint satisfaction: a framework for decision problems under incomplete knowledge. In *Proceedings of the 12th National Conference on A.I. (AAAI-96)*, pages 175–180, Portland (Oregon, USA).

[Ghallab and Vidal, 1995] Ghallab, M. and Vidal, T. (1995). Focusing on a sub-graph for managing efficiently numerical temporal constraints. In *FLorida A.I. Research Symposium (FLAIRS-95)*, Melbourne Beach (FL, USA).

[Jourdan et al., 1997] Jourdan, M., Layaïda, N., and Sabry-Ismail, L. (1997). Time representation and management in MADEUS: an authoring environment for multimedia documents. In Freeman, M., Jardetzky, P., and Vin, H., editors, *Multimedia Computing and Networking*, pages 68–79.

[Mackworth and Freuder, 1985] Mackworth, A. and Freuder, E. (1985). The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25(1):65–74.

[Maler et al., 1995] Maler, O., Pnueli, A., and Sifakis, J. (1995). On the synthesis of discrete controllers for timed systems. In *Proceedings of the 12th Symposium on Theoretical Aspects of Computer Science*, München (Germany).

[Meiri, 1991] Meiri, I. (1991). Combining qualitative and quantitative constraints in temporal reasoning. In *Proceedings of the 9th National Conference on A.I. (AAAI-91)*, pages 260–267, Anaheim (CA, USA).

[Morris and Muscettola, 1999] Morris, P. and Muscettola, N. (1999). Managing temporal uncertainty through waypoint controllability. In Dean, T., editor, *Proceedings of the 16th International Joint Conference on A.I. (IJCAI-99)*, pages 1253–1258, Stockholm (Sweden). Morgan Kaufmann.

[Morris et al., 1998] Morris, P., Muscettola, N., and Tsamardinos, I. (1998). Reformulating temporal plans for efficient execution. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR-98)*, Trento (Italy).

[Schwalb and Dechter, 1997] Schwalb, E. and Dechter, R. (1997). Processing disjunctions in temporal constraint networks. *Artificial Intelligence*, 93:29–61.

[Vidal, 2000] Vidal, T. (2000). A unified and dynamic approach for dealing with temporal uncertainty and conditional planning. In *Proceedings of the 5th International Conference on Artificial Intelligence Planning & Scheduling (AIPS2000)*, Breckenridge (Co, USA).

[Vidal and Fargier, 1999] Vidal, T. and Fargier, H. (1999). Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental & Theoretical Artificial Intelligence*, 11:23–45.

[Vidal and Ghallab, 1996] Vidal, T. and Ghallab, M. (1996). Dealing with uncertain durations in temporal constraint networks dedicated to planning. In *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI-96)*, pages 48–52, Budapest (Hungary).

[Vila and Reichgelt, 1993] Vila, L. and Reichgelt, H. (1993). The token reification approach to temporal reasoning. Technical Report 1/93, Dept of Computer Science, UWI.

[Vilain et al., 1989] Vilain, M., Kautz, H., and van Beek, P. (1989). Constraint propagation algorithms: a revised report. In *Readings in Qualitative Reasoning about Physical Systems*. Morgan Kaufman.

[Yovine, 1997] Yovine, S. (1997). Kronos: a verification tool for real-time systems. *International Journal of Software Tools for Technology Transfer*, 1(1/2).

The Augmented Interval and Rectangle Networks

Jean-François Condotta

Institut de recherche en informatique de Toulouse (IRIT)
118 route de Narbonne, F-31062 Toulouse Cedex
e-mail: condotta@irit.fr

Abstract

We augment Allen's Interval Algebra networks and Rectangle Algebra networks by quantitative constraints represented by $\mathcal{STP}$s. With the help of polynomial algorithms based on the traditional and weak path-consistency methods, we prove the tractability of the consistency problem of preconvex augmented interval networks and strongly-preconvex augmented rectangle networks.

Keywords: Temporal and spatial reasoning, Interval Algebra and Rectangle Algebra, constraint networks, complexity.

1 Introduction

Temporal and spatial reasoning with constraints is a relevant activity in artificial intelligence. Concerning qualitative temporal reasoning, Allen's Interval Algebra [1] (IA) is one of the most known and used formalisms. Allen takes intervals as primitive temporal entities and considers 13 atomic relations between these intervals (fig. 1). These relations represent all the possible relative positions between two intervals on the rational line. With this formalism we can represent qualitative constraints about relative positions of a set of temporal entities such as: "a temporal entity must be realized before or during another temporal entity". On the other hand, to represent quantitative (*i.e. numeric*) temporal constraints such as: "an event must be realized between 8h and 11h a.m.", structures based on inequations and inequalities on rational or real variables representing temporal-points were defined [6, 7, 11]. To take advantage of both kinds of formalisms (qualitative and quantitative) researchers augmented IA with quantitative constraints on interval endpoints. The structures hence obtained inherit the main drawback of IA: the consistency problem of a set of constraints is a NP-complete problem.

In this paper, we augment Allen's structure simply by allowing quantitative constraints defined by a Simple Temporal Problem ($\mathcal{STP}$). With a polynomial algorithm we show that the consistency problem of a set of constraints whose qualitative part is a set of preconvex constraints is polynomial. This result subsumes the previous complexity results of Meiri [14] and Kautz *et al.* [10]. The polynomial methods presented by Koubarakis [11] and Jonsson *et al.* [9] to solve systems of inequality and inequation disjunctions can also be used to solve this problem. But the implementation of these methods is very difficult because they are based on the Kachian's algorithm (used to solve the linear programming problem). Moreover these methods are known to be polynomial but their exact complexity is unknown. For this reason we claim that the polynomial algorithm presented in this paper is very interesting.

Concerning qualitative spatial reasoning, Rectangle Algebra (RA) is a spatial formalism which extends IA to dimension 2. The objects considered are the rectangles of the plane whose sides are parallel to the axes of some orthogonal basis in a 2-dimensional Euclidean space. The atomic relations between these rectangles are defined from the "Cartesian product" of IA atomic relations. This structure can be particularly useful in areas such as architecture [4], or image synthesis [12] where the user needs, for some applications, to define position constraints on objects represented by rectangles. Similarly to IA, RA allows uniquely to express qualitative constraints. To remedy that we augment this structure by allowing also quantitative constraints about the distances between the endpoints of the projections of the rectangles onto the axes. Hence we can express constraints such as: "the length of a rectangle side is between 5 and 6 m". We show that the consis-

tency problem of a set of constraints whose qualitative constraints are strongly-preconvex relations is polynomial.

The following section is devoted to some recalls about IA and RA. Section 3 is dedicated to some tractable fragments of IA and RA for the consistency problem. Section 4 is about $\mathcal{STP}$ and $\mathcal{STP}^{\neq}$, with also some "bridges" between the $\mathcal{STP}$ formalism and Interval Algebra. We introduce augmented interval and rectangle networks in section 5. In section 6, we characterize two tractable cases for the consistency problem of these constraint networks: the preconvex augmented interval networks and the strongly-preconvex augmented rectangle networks.

2 Interval Algebra and Rectangle Algebra

Interval Algebra was introduced by Allen [1] for qualitative temporal reasoning. IA is a relation algebra based on 13 atomic relations:

$$\mathcal{B}_{int} = \{b, m, o, s, d, f, bi, mi, oi, si, di, fi, eq\}$$

which constitute the exhaustive list of the relations which can be satisfy between two rational intervals (fig. 1 (a)). We will denote by X^- (resp. X^+) the lower endpoint (resp. the upper endpoint) of the interval X.

Rectangle Algebra (RA) [8, 15, 2, 3] is an extension of IA. The objects considered by RA are the rectangles whose sides are parallel to the axes of some orthogonal basis of the 2-dimensional Euclidean space. Given a rectangle x, X_1 (resp. X_2) will denote the projection of x onto the first axis (resp. onto the second axis). The set of the atomic relations between these objects is defined from the IA one by the following way: $\mathcal{B}_{rec} = \{(A, B) : A, B \in \mathcal{B}_{int}\}$. Two rectangles x and y satisfy the atomic relation (A, B) iff X_1 and Y_1 satisfy A and X_2 and Y_2 satisfy B. In figure 1 (b) are represented two rectangles x and y satisfying the atomic relation (m, b).

The set of IA relations (resp. RA relations) is the power set of $\mathcal{B}_{int}$ (resp. $\mathcal{B}_{rec}$), *i.e.* $2^{\mathcal{B}_{int}}$ (resp. $2^{\mathcal{B}_{rec}}$). Each relation can be seen as the disjunction of the atomic relations forming it. Given X and Y two rational intervals and $R \in 2^{\mathcal{B}_{int}}$, $X\ R\ Y$ will denote that X and Y satisfy one of the atomic relations of R. We will use the same notation for RA. Given R a relation of AR, R_1 (resp. R_2) is the interval relation $\{A : (A, B) \in R\}$ (resp. $\{B : (A, B) \in R\}$). The sets $2^{\mathcal{B}_{int}}$ and $2^{\mathcal{B}_{rec}}$ are equipped with the following fundamental operations: intersection ($\cap$), union ($\cup$), composition ($\circ$) and inverse ($^{-1}$). The intersection and union operations are the homonymous set operations. The composition of two relations can be defined with the composition of the atomic relations which make it up: $R \circ S = \cup\{A \circ B : A \in R \text{ and } B \in S\}$. Allen describes the composition of IA atomic relations in a table from which the composition of any pair of relations is easily deducible. The composition of two RA relations is deduced from the one of IA: let $(A, B), (C, D) \in \mathcal{B}_{rec}$ be, $(A, B) \circ (C, D) = (A \circ C) \times (B \circ D)$. The inverse of a relation belonging to $2^{\mathcal{B}_{int}}$ ($2^{\mathcal{B}_{rec}}$) is the set of the inverse of its atomic relations. The inverse of an atomic relation (A, B) of $\mathcal{B}_{rec}$ is (A^{-1}, B^{-1}).

Qualitative temporal information between several intervals is represented by a particular binary constraint network: an interval network. An interval network $\mathcal{N}$ is a structure (V, C) where V is a set of variables $V_1, \ldots, V_{|V|}$ representing intervals and where C is a mapping from $V \times V$ onto $2^{\mathcal{B}_{int}}$ corresponding to the binary constraints between the intervals. We will denote by C_{ij} the relation $C(V_i, V_j)$. C is such that: $\forall i \in 1, \ldots, |V|$, $C_{ii} = \{eq\}$ and $\forall i, j \in 1, \ldots, |V|$, $C_{ij} = C_{ji}^{-1}$. C_{ij} represents the set of the positions allowed between the two intervals represented by V_i and V_j. A rectangle network is defined in the same way excepted that V is a set of variables representing rectangles and C is a mapping onto $2^{\mathcal{B}_{rec}}$ such that $C_{ii} = \{(eq, eq)\}$.

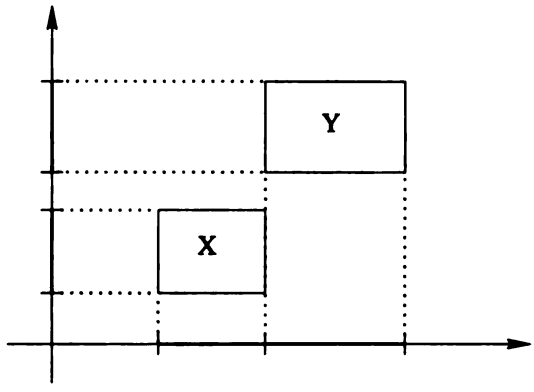
The projections of a rectangle network $\mathcal{N} = (V, C)$ are the interval networks $\mathcal{N}_1 = (V', C')$ and $\mathcal{N}_2 = (V'', C'')$ with $V = V' = V''$, $\forall i, j \in 1, \ldots, |V|$, $C'_{ij} = (C_{ij})_1$ and $C''_{ij} = (C_{ij})_2$. $\mathcal{N}_1$ (resp. $\mathcal{N}_2$) corresponds to the constraints implied by $\mathcal{N}$ on the projections of the rectangles of V onto the first (resp. the second) axis.

A subnetwork $\mathcal{N}' = (V', C')$ of a network $\mathcal{N} = (V, C)$ (expressed by $\mathcal{N}' \subseteq \mathcal{N}$) is a network such that $V' = V$ and $\forall i, j \in 1, \ldots, |V'|$, $C'_{ij} \subseteq C_{ij}$. Moreover if we have a strict inclusion for a pair (i, j), $\mathcal{N}'$ is a strict subnetwork of $\mathcal{N}$ ($\mathcal{N}' \subset \mathcal{N}$).

An instantiation of a rectangle network (resp. an interval network) $\mathcal{N} = (V, C)$ is a mapping m which associates each variable $V_i \in V$ with a rational rectangle (resp. a rational interval), denoted by m_i. m_{ij} will denote the atomic relation satisfied by m_i and m_j. m is a consistent instantiation of $\mathcal{N}$ iff $\forall i, j \in 1, \ldots, |V|$, $m_{ij} \in Cij$. A network is consistent iff it admits a consistent instantiation. Two rectangle networks or interval networks with the same variables are equivalent iff they have the same consistent instantiations. A network $\mathcal{N}$ is minimal iff every atomic relation of its constraints can be satisfied by a consistent instantiation. The minimal network of a network is the smallest equivalent subnetwork.

Relation	Symbol	Reverse	Meaning	Dim
precedes	p	pi		2
meets	m	mi		1
overlaps	o	oi		2
starts	s	si		1
during	d	di		2
finishes	f	fi		1
equals	eq	eq		0

(a)



(b)

Figure 1: (a) the set of the 13 atomic relations of IA, (b) the atomic relation (m, b).

3 A NP-Complete Problem

Given an interval or a rectangle network the main problem is to know whether this network is consistent. This problem is NP-complete. The goal of numeral studies was to find particular fragments of relations for which these problems are polynomial. In this section we focus on some tractable fragments found: the sets of the convex relations [17, 5] and preconvex relations [13, 16] of IA, the sets of the convex relations [2] and strongly-preconvex relations [3] of RA.

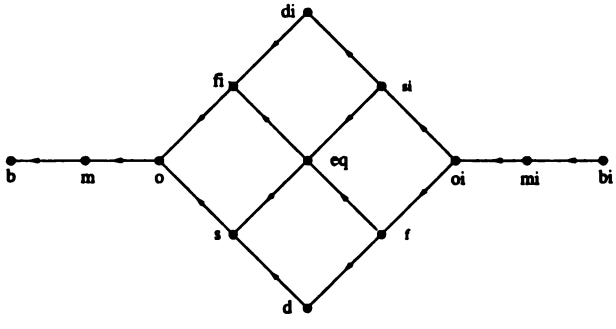


Figure 2: The interval lattice $(\mathcal{B}_{int}, \leq)$.

Ligozat [13] arranges in order the IA atomic relations and obtains the interval lattice $(\mathcal{B}_{int}, \leq)$ (see fig. 2). The convex relations of IA correspond to the intervals of this lattice. The convex relations of RA [2] can be easily characterized: a relation $R \in 2^{\mathcal{B}_{rec}}$ is convex iff $R = R_1 \times R_2$ and R_1, R_2 are two convex relations of $2^{\mathcal{B}_{int}}$. The convex closure of a relation R belonging to $2^{\mathcal{B}_{int}}$ (resp. $2^{\mathcal{B}_{rec}}$), denoted by $I(R)$, is the smallest convex relation of $2^{\mathcal{B}_{int}}$ (resp. $2^{\mathcal{B}_{rec}}$) which contains R.

For example, $\{b, m, o\}$ is a convex relation corresponding to the interval $[b, o]$, whereas the relation $\{b, o\}$ is not convex. Moreover, $I(\{b, m, o\}) = I(\{b, o\}) = \{b, m, o\}$.

The dimension of an atomic relation $A \in \mathcal{B}_{int}$, denoted by $dim(A)$, can be defined as being the difference between the number 2 and the number of endpoint equalities forced by A. The greater the dimension of an atomic relation is, the less constrain we are to impose the same values to the interval endpoints. The dimension of an atomic relation (A, B) of $\mathcal{B}_{rec}$ is defined in the following way: $dim((A, B)) = dim(A) + dim(B)$. Let $R \in 2^{\mathcal{B}_{int}}$ or $2^{\mathcal{B}_{rec}}$, $dim(R) = max\{dim(A) : A \in R\}$. For illustration, we have $dim(b) = dim(o) = 2 - 0 = 2$ and $dim(eq) = 2 - 2 = 0$, hence $dim(\{b, o, eq\}) = 2$ and $dim(\{(b, eq), (o, b), (eq, o)\}) = 4$.

Given an interval or a rectangle network $\mathcal{N} = (V, C)$ and a consistent instantiation m of this network, m will be maximal for $\mathcal{N}$ iff $\forall i, j \in 1, \ldots, |V|$, $dim(m_{ij}) = dim(C_{ij})$.

A relation R of $2^{\mathcal{B}_{int}}$ is preconvex iff $dim(I(R) \setminus R) < dim(R)$. A relation R of $2^{\mathcal{B}_{rec}}$ is strongly-preconvex iff for each convex relation S of RA, $dim(I(R \cap S) \setminus (R \cap S)) < dim(R \cap S)$.

For example, the interval relation $\{b, o\}$ is preconvex, whereas $\{(d, d), (s, s), (s, eq)\}$ is a strongly-preconvex relation of $2^{\mathcal{B}_{rec}}$.

A convex (resp. preconvex, strongly-preconvex) network is a network which has only convex (resp. preconvex, strongly-preconvex) relations as constraints. Given an interval or a rectangle network $\mathcal{N} = (V, C)$, the convex closure of $\mathcal{N}$, denoted by $I(\mathcal{N})$, is the network (V', C') such that $V' = V$ and $\forall i, j \in 1, \ldots, |V|$, $C'_{ij} = I(C_{ij})$.

It is time now to recall some properties proved in [5, 13, 16, 2, 3].

Fact 1

(a) Every convex relation of IA, resp. of RA, is preconvex, resp. strongly-preconvex.

(b) The sets of the convex relations of IA, of the preconvex relations of IA and of the convex relations of RA are stable for $\circ$, $\cap$ and $^{-1}$. The set of the strongly-preconvex relations of RA is stable for $\cap$ and $^{-1}$.

Before going on, let us remind the definition of the path-consistency and weak path-consistency: an interval or a rectangle network $\mathcal{N} = (V, C)$ is path-consistent (resp. weakly path-consistent) iff $\forall i, j, k \in 1, \ldots, |V|$, $C_{ij} \subseteq C_{ik} \circ C_{kj}$ (resp. $C_{ij} \subseteq I(C_{ik} \circ C_{kj})$) and $C_{ij} \neq \{\}$. The path-consistency (resp. weak path-consistency) method consists in transforming a network $\mathcal{N} = (V, C)$ in an equivalent network which is path-consistent (resp. weakly path-consistent), or which contains the empty relation, by iterating the triangulation operation: $C_{ij} \leftarrow C_{ij} \cap (C_{ik} \circ C_{kj})$ (resp. $C_{ij} \leftarrow C_{ij} \cap I(C_{ik} \circ C_{kj})$), for every triple $i, j, k \in 1, \ldots, |V|$, until a fixed point is reached. These methods can be implemented in $O(|V|^3)$. As one can guess from its complexity, the (weak) path-consistency method despite its soundness is not complete. Indeed, after (weak) path-consistency method application, if the network contains the empty relation then it is inconsistent else its consistency is not sure.

Fact 2 *Let $\mathcal{N}$ be an interval or rectangle network.*

(a) If $\mathcal{N}$ is weakly path-consistent then $I(\mathcal{N})$ is path-consistent, moreover, if $\mathcal{N}$ is a rectangle network then $I(\mathcal{N}_1)$ and $I(\mathcal{N}_2)$ are two path-consistent networks.

(b) If $\mathcal{N}$ is convex and path-consistent then $\mathcal{N}$ admits a maximal consistent instantiation.

(c) If $\mathcal{N}$ is a preconvex interval network or a strongly-preconvex rectangle network and if it is weakly path-consistent then every maximal consistent instantiation of $I(\mathcal{N})$ is a maximal consistent instantiation of $\mathcal{N}$.

(d) If $\mathcal{N}$ is a weakly path-consistent strongly-preconvex rectangle network then from every pair of maximal consistent instantiations m' of $I(\mathcal{N})_1$ and m'' of $I(\mathcal{N})_2$, we can construct a maximal consistent instantiation m of $\mathcal{N}$ [1].

[1] m'_i (resp. m''_i) corresponds to the projection of m_i onto the first (resp. the second) axis.

From all this, it follows that concerning the consistency problem, the path-consistency method and the weak path-consistency method are complete for the preconvex networks of IA and for the strongly-preconvex networks of RA.

4 Simple Temporal Problems

This section is devoted to quantitative constraint networks between points: the $\mathcal{STP}$s [6] and the $\mathcal{STP}^{\neq}$s [7]. A $\mathcal{STP}$ $\mathcal{S}$ is a structure (V, C) in which V is a set $V_1, \ldots, V_{|V|}$ of variables representing points and in which C is a mapping from $V \times V$ onto a set of (closed or open) intervals (with finite or infinite bounds) defining quantitative binary constraints between points. C_{ij} contains the values allowed for $V_j - V_i$. The constraint between a variable and itself is $[0, 0]$, and C_{ij} is the interval formed by the opposed values of those belonging to C_{ji}.
A $\mathcal{STP}^{\neq}$ is slightly different from a $\mathcal{STP}$, for a $\mathcal{STP}^{\neq}$, every constraint C_{ij} between two points V_i and V_j is an interval I_{ij} minus a set E_{ij} containing a finite number of values belonging to I_{ij}. C_{ij} expresses the constraint $V_j - V_i \in I_{ij} \setminus E_{ij}$. The relaxed network of a $\mathcal{STP}^{\neq}$ $\mathcal{S} = (V, C)$ is the $\mathcal{STP}$ $\mathcal{S}' = (V, C')$ defined by: $\forall i, j \in 1, \ldots, |V|$, $C'_{ij} = I_{ij}$ with $C_{ij} = I_{ij} \setminus E_{ij}$.
The operations intersection ($\cap$), composition ($\circ$) and inverse ($^{-1}$) are also defined for the intervals ($\mathcal{STP}$ constraints) [6]. Briefly, let us recall that the intersection of two intervals is the usual set intersection, the inverse of an interval is the interval whose lower (resp. upper) bound is the opposite of the upper (resp. the lower) bound of the interval considered. The composition of two intervals is the interval whose lower (resp. upper) bound is the sum of the lower (resp. the upper) bounds of the two intervals considered. It satisfies the following property:

Lemma 1 *Let I, J, I', J' be intervals, if $I \subseteq I'$ and $J \subseteq J'$ then $I \circ J \subseteq I' \circ J'$.*

The notions of subnetwork, minimality, instantiation, consistency, path-consistency defined previously for qualitative networks are still valid for $\mathcal{STP}$s and $\mathcal{STP}^{\neq}$s. Let us add that a $\mathcal{STP}$ $\mathcal{S} = (V, C)$ entails $V_j - V_i = a$, with $V_j, V_i \in V$ and a a rational iff $\mathcal{S}$ does not admit a consistent instantiation m such that $m_j - m_i \neq a$. The consistency problem of a $\mathcal{STP}$ is polynomial and can be solved by the path-consistency method:

Fact 3 *Let $\mathcal{S}$ be a $\mathcal{STP}$. If $\mathcal{S}$ is path-consistent then $\mathcal{S}$ is minimal.*

Concerning the $\mathcal{STP}^{\neq}$s, Gerevini *et al.* [7] prove the following result:

Fact 4 *Let $\mathcal{S} = (V, C)$ be a $\mathcal{STP}^{\neq}$ and let $\mathcal{S}'$ be the relaxed $\mathcal{STP}$ of $\mathcal{S}$. $\mathcal{S}$ is consistent iff $\mathcal{S}'$ is consistent and $\forall i, j \in 1, \ldots, |V|$, with $C_{ij} = C'_{ij} \setminus E_{ij}$, if $a \in E_{ij}$ then $\mathcal{S}'$ does not entail $V_j - V_i = a$.*

A $\mathcal{STP}$ $\mathcal{S} = (V, C)$ does not entail $V_j - V_i = a$ iff the minimal equivalent $\mathcal{STP}$ of $\mathcal{S}$ does not have the constraint $[a, a]$ between V_i and V_j. So, from fact 3 we have:

Lemma 2 *Let $\mathcal{S} = (V, C)$ be a $\mathcal{STP}^{\neq}$ and let $\mathcal{S}' = (V', C')$ be its relaxed network. If $\mathcal{S}'$ is path-consistent and if $C'_{ij} = [a, a]$ (with a being a rational) implies that $a \notin E_{ij}$, for all $i, j \in 1, \ldots, |V|$, with $C_{ij} = C'_{ij} \setminus E_{ij}$, then $\mathcal{S}$ is consistent.*

Hence, to check the consistency of a $\mathcal{STP}^{\neq}$ $\mathcal{S}$ we can apply the path-consistency method on its relaxed network $\mathcal{S}'$ and check that the singleton intervals of $\mathcal{S}'$ does not correspond to excluded values of $\mathcal{S}$. Gerevini *et al.* [7] define an algorithm building a consistent instantiation of a consistent $\mathcal{STP}^{\neq}$ in $O(n^3)$, with n the number of variables, we will call it SOLUTION-STP$^{\neq}$. Let the set $\mathcal{E}_{0\infty}$=$\{]-\infty, +\infty[,]-\infty, 0[,]-\infty, 0],]0, +\infty[, [0, +\infty[, [0, 0]\}$ be. Given an interval I, we define the extension of I, noted $\mathsf{EXT}(I)$, by the smallest interval belonging to $\mathcal{E}_{0\infty}$ and containing I ($\mathsf{EXT}(\emptyset) = \emptyset$). We extend this notion to the $\mathcal{STP}$s:

Definition 1 *Let $\mathcal{S} = (V, C)$ be a $\mathcal{STP}$. The extension of $\mathcal{S}$, denoted by $EXT(\mathcal{S})$, corresponds to the $\mathcal{STP}$ (V, C') defined by: $\forall i, j \in 1, \ldots, |V|$, C'_{ij}=$EXT(C_{ij})$.*

We can establish the following properties:

Lemma 3 *Let $\mathcal{S} = (V, C)$ and $\mathcal{S}'$ be two $\mathcal{STP}$s. (a) If $\mathcal{S} \subseteq \mathcal{S}'$ then $EXT(\mathcal{S}) \subseteq EXT(\mathcal{S}')$. (b) $\mathcal{S} \subseteq EXT(\mathcal{S})$. (c) If $\forall i, j \in 1, \ldots, |V|$, $C_{ij} \in \mathcal{E}_{0\infty}$ then $EXT(\mathcal{S}) = \mathcal{S}$.*

From lemmas 1 and 3 we can prove the following result:

Lemma 4 *If $\mathcal{S}$ is a path-consistent $\mathcal{STP}$ then $EXT(\mathcal{S})$ is path-consistent.*

Proof For each triple $i, j, k \in 1, \ldots, |V|$, we have $C_{ik} \subseteq \mathsf{EXT}(C_{ik})$ and $C_{kj} \subseteq \mathsf{EXT}(C_{kj})$. Consequently, $C_{ik} \circ C_{kj} \subseteq \mathsf{EXT}(C_{ik}) \circ \mathsf{EXT}(C_{kj})$ (lemma 1). As $\mathcal{S}$ is path-consistent, $C_{ij} \subseteq C_{ik} \circ C_{kj}$. From all this, it follows that $C_{ij} \subseteq \mathsf{EXT}(C_{ik}) \circ \mathsf{EXT}(C_{kj})$. From lemma 3 (a) we deduce that $\mathsf{EXT}(C_{ij}) \subseteq \mathsf{EXT}(\mathsf{EXT}(C_{ik}) \circ \mathsf{EXT}(C_{kj}))$. We have $\mathsf{EXT}(C_{ik})$ and $\mathsf{EXT}(C_{kj}) \in \mathcal{E}_{0\infty}$, it follows that $\mathsf{EXT}(C_{ik}) \circ \mathsf{EXT}(C_{kj}) \in \mathcal{E}_{0\infty}$. Hence, $\mathsf{EXT}(\mathsf{EXT}(C_{ik}) \circ \mathsf{EXT}(C_{kj})) = \mathsf{EXT}(C_{ik}) \circ \mathsf{EXT}(C_{kj})$ (lemma 3 (c)) and $\mathsf{EXT}(C_{ij}) \subseteq \mathsf{EXT}(C_{ik}) \circ \mathsf{EXT}(C_{kj})$. Moreover, as $C_{ij} \neq \emptyset$ $\mathsf{EXT}(C'_{ij})$ is not empty. ⊣

The sequel of this section is devoted to "bridges" between interval networks and $\mathcal{STP}$s. A constraint between two intervals X and Y defined by a convex relation of IA can be expressed by a set of constraints on the bounds of X and Y with relations of Point Algebra (PA) [10]: $\{\leq, <, >, \geq, =, all\}$. Van Beek *et al.* [5] gave a table providing for each convex relation of IA its translation in PA (we call this table "the van Beek's table"). Every qualitative constraint using a relation of the set $\{\leq, <, >, \geq, =, all\}$ can be expressed equivalently by a $\mathcal{STP}$ constraint (see table 1). Given a set V of intervals, $Points(V)$ will denote the set of the bounds of the intervals in V. We define a translation from a convex network of IA to an equivalent $\mathcal{STP}$ by the following way:

Definition 2 *Let $\mathcal{N} = (V, C)$ be a convex interval network. $STP(\mathcal{N})$ is the $\mathcal{STP}$ $\mathcal{S} = (Points(V), C')$ where C' corresponds to the quantitative constraints between the bounds of the intervals in V, which are equivalent to the ones of C and built from van Beek's table and table 1.*

The STP translation of the IA atomic relations is given by table 2. We note that for each one of these the constraints given in this table form a path-consistent $\mathcal{STP}$ on the variables X^-, X^+, Y^- and Y^+. From this table we can calculate the translation of every convex relations R of IA. The quantitative constraint between two bounds in the translation of R is the union of the quantitative constraints between these two bounds which are in the translations of the atomic relations forming R.

For example, X $\{o, s, d\}$ Y corresponds to the quantitative constraints: $C_{X^-Y^-} =]0, +\infty[\cup [0, 0] \cup]-\infty, 0[=]-\infty, +\infty[$, $C_{X^-Y^+} =]0, +\infty[\cup]0, +\infty[\cup]0, +\infty[=]0, +\infty[$, *etc* .

Now, we are going to define the inverse translation of STP: IN. Any $\mathcal{STP}$ does not correspond to a (convex) interval network, we restrict the considered $\mathcal{STP}$s:

Lemma 5 *Let V' be a set of intervals and let $\mathcal{S} = (V = Points(V'), C)$ be a $\mathcal{STP}$ such that $\forall i, j \in 1, \ldots, |V|$, $C_{ij} \in \mathcal{E}_{0\infty}$ and $C_{ij} =]0, +\infty[$ if V_i and V_j are the lower and the upper bound of an interval in V. If $\mathcal{S}$ is path-consistent then there exists one and only one convex interval network $\mathcal{N} = (V', C')$ such that $STP(\mathcal{N}) = \mathcal{S}$, we will denote this network by $IN(\mathcal{S})$.*

Proof Let $X = (X^-, X^+)$ and $Y = (Y^-, Y^+)$ be two interval variables of V'. On the one hand

Qualitative	Quantitative	Qualitative	Quantitative
$X\ \{<\}\ Y$	$Y-X \in]0,+\infty[$	$X\ \{>\}\ Y$	$Y-X \in]-\infty,0[$
$X\ \{<,=\}\ Y$	$Y-X \in [0,+\infty[$	$X\ \{>,=\}\ Y$	$Y-X \in]-\infty,0]$
$X\ \{=\}\ Y$	$Y-X \in [0,0]$	$X\ \{>,=,<\}\ Y$	$Y-X \in]-\infty,+\infty[$

Table 1: Translation between qualitative constraints and quantitative constraints.

C_{XY}	$C_{X^-Y^-}$	$C_{X^-Y^+}$	$C_{X^+Y^-}$	$C_{X^+Y^+}$	C_{XY}	$C_{X^-Y^-}$	$C_{X^-Y^+}$	$C_{X^+Y^-}$	$C_{X^+Y^+}$
$\{oi\}$	$]-\infty,0[$	$]0,+\infty[$	$]-\infty,0[$	$]-\infty,0[$	$\{b\}$	$]0,+\infty[$	$]0,+\infty[$	$]0,+\infty[$	$]0,+\infty[$
$\{m\}$	$]0,+\infty[$	$]0,+\infty[$	$[0,0]$	$]0,+\infty[$	$\{bi\}$	$]-\infty,0[$	$]-\infty,0[$	$]-\infty,0[$	$]-\infty,0[$
$\{mi\}$	$]-\infty,0[$	$[0,0]$	$]-\infty,0[$	$]-\infty,0[$	$\{d\}$	$]-\infty,0[$	$]0,+\infty[$	$]-\infty,0[$	$]0,+\infty[$
$\{s\}$	$[0,0]$	$]0,+\infty[$	$]-\infty,0[$	$]0,+\infty[$	$\{di\}$	$]0,+\infty[$	$]0,+\infty[$	$]-\infty,0[$	$]-\infty,0[$
$\{si\}$	$[0,0]$	$]0,+\infty[$	$]-\infty,0[$	$]-\infty,0[$	$\{o\}$	$]0,+\infty[$	$]0,+\infty[$	$]-\infty,0[$	$]0,+\infty[$
$\{f\}$	$]-\infty,0[$	$]0,+\infty[$	$]-\infty,0[$	$[0,0]$	$\{fi\}$	$]0,+\infty[$	$]0,+\infty[$	$]-\infty,0[$	$[0,0]$
$\{eq\}$	$[0,0]$	$]0,+\infty[$	$]-\infty,0[$	$[0,0]$					

Table 2: Translation of $\mathcal{B}_{int}$ in quantitative constraints ($C_{X^-X^+} = C_{Y^-Y^+} =]0,+\infty[$).

we know $\mathcal{S}$ is path-consistent, and on the other hand the constraints of C belong to the set $\mathcal{E}_{0\infty}$ and $C_{X^-X^+} = C_{Y^-Y^+} =]0,+\infty[$. By examining the exhaustive list of all the possible constraint configurations between the points X^-, X^+, Y^- and Y^+ by computer program, we notice that all those configurations correspond to a convex interval relation translation by the STP translation. ⊣

With a similar proof we can prove the following result:

Lemma 6 *Let V'' be a set of intervals and let $\mathcal{S} = (V = Points(V''), C)$, $\mathcal{S}' = (V, C')$ be two path-consistent $\mathcal{STP}$s such that $\forall i,j \in 1,\ldots,|V|$, $C_{ij} \in \mathcal{E}_{0\infty}$ and if V_i and V_j are the lower and the upper bounds of an interval in V'' then $C_{ij} =]0,+\infty[$. If $\mathcal{S} \subseteq \mathcal{S}'$ (resp. $\mathcal{S} \subset \mathcal{S}'$) then $IN(\mathcal{S}) \subseteq IN(\mathcal{S}')$ (resp. $IN(\mathcal{S}) \subset IN(\mathcal{S}')$).*

5 The augmented networks

With interval networks we can reason on relative positions of intervals in a qualitative way ; only the order between the bounds of the intervals is relevant. Now, we would like to add quantitative constraints about the distance between the interval bounds. For that purpose we introduce a simple and sufficient structure:

Definition 3 *An augmented interval network $\mathcal{M}$ is a pair $(\mathcal{N}, \mathcal{S})$ where $\mathcal{N} = (V, C)$ is an interval network which represents the qualitative constraints on the intervals in V and $\mathcal{S} = (Points(V), C')$ is a $\mathcal{STP}$ representing the quantitative constraints on the distances between the bounds of the intervals in V.*

In the past, similar approaches using both qualitative and quantitative constraints were defined [14, 10]. Our approach is close to Kautz *et al.*'s one, indeed to be clearer, we use two distinct constraint networks, one which containing the qualitative constraints, the other one containing the quantitative constraints.
Like for the intervals, to take into account quantitative constraints on the lengths between the sides of the rectangles, we define the augmented rectangle networks:

Definition 4 *An augmented rectangle network $\mathcal{M}$ is a triple $(\mathcal{N}, \mathcal{S}_1, \mathcal{S}_2)$ where $\mathcal{N} = (V, C)$ is a rectangle network which represents the qualitative constraints on the rectangles in V. $\mathcal{S}_1 = (Points(V'), C')$ and $\mathcal{S}_2 = (Points(V''), C'')$ are two $\mathcal{STP}$s representing the quantitative constraints on the distances between the interval bounds of respectively V' and V'', where V' (resp. V'') represents the set of the projections of the rectangles in V onto the first axis (resp. the second axis).*

An augmented network is convex (resp. strongly-preconvex, preconvex) if its qualitative network is convex (resp. strongly-preconvex, preconvex). A consistent instantiation m of an augmented network is a consistent instantiation of its qualitative network and of its $\mathcal{STP}$(s). Moreover, m is maximal if it is a maximal instantiation of the qualitative network of the augmented network.
The consistency problem of an augmented network is NP-complete in the general case. In the next section, we characterize two polynomial cases.

6 Tractable cases

We know that every path-consistent $\mathcal{STP}$ admits a consistent instantiation, now we focus on particular consistent instantiations of these $\mathcal{STP}$s: "the instantiations least satisfying the value 0 in the constraint values":

Lemma 7 *Let $\mathcal{S} = (V, C)$ be a path-consistent $\mathcal{STP}$. There exists a consistent instantiation m of $\mathcal{S}$ such that $\forall i, j \in 1, \ldots, |V|$, $m_j - m_i = 0$ iff $C_{ij} = [0, 0]$.*

Proof Let the $\mathcal{STP}^{\neq}$ $\mathcal{S}' = (V, C')$ be defined by: $\forall i, j \in 1, \ldots, |V|$, $C'_{ij} = I_{ij} \setminus E_{ij}$, with $I_{ij} = C_{ij}$ and $E_{ij} = \{0\}$ if $0 \in C_{ij}$ and $C_{ij} \neq [0, 0]$. $\mathcal{S}$ is the relaxed network of $\mathcal{S}'$. As $\mathcal{S}$ is path-consistent and since $\forall i, j \in 1, \ldots, |V|$, if $a \in E_{ij}$ (in fact a is necessarily 0) then $I_{ij} \neq [a, a]$, we deduce that $\mathcal{S}'$ admits a consistent instantiation m calculable in polynomial time (lemma 2). We note that m is a consistent instantiation of $\mathcal{S}$ and that $\forall i, j \in 1, \ldots, |V|$, $m_j - m_i = 0$ iff $C_{ij} = [0, 0]$. ⊣

From the previous lemma we can prove the following property:

Lemma 8 *Let $\mathcal{M} = (\mathcal{N}, \mathcal{S})$ be a convex augmented interval network. If $STP(\mathcal{N}) = EXT(\mathcal{S})$ and $\mathcal{S}$ is path-consistent then $\mathcal{M}$ admits a maximal consistent instantiation.*

Proof As $\mathcal{S} = (V' = Points(V), C')$ is path-consistent, there exists a consistent instantiation m of $\mathcal{S}$ such that m assigns the same value to two bounds X and $Y \in V'$ iff $C'_{XY} = [0, 0]$ (lemma 7). $\mathcal{S} \subseteq \mathsf{EXT}(\mathcal{S})$ and $\mathsf{EXT}(\mathcal{S}) = \mathsf{STP}(\mathcal{N})$, it follows that m is also a consistent instantiation of $\mathcal{N} = (V, C)$. Let us show that m is a maximal for $\mathcal{N}$. Let V_i and V_j be two intervals of V, and let m_{ij} be the atomic relation in C_{ij} satisfied by m. Let us suppose that there exists $A \in C_{ij}$ such that $dim(A) > dim(m_{ij})$. Hence, m_{ij} forces at least one equality between two bounds X and Y of respectively V_i and V_j which is not forced by A. By denoting the constraints of $\mathsf{STP}(\mathcal{N})$ by C'', it follows that C''_{XY} is among the following intervals: $[0, +\infty[$, $]-\infty, 0]$ or $]-\infty, +\infty[$. This implies that $C'_{XY} \neq [0, 0]$. However m satisfies m_{ij} between V_i and V_j, it follows that m assigns to X and Y the same value and so $C'_{XY} = [0, 0]$. There is a contradiction. Consequently $dim(m_{ij}) = dim(C_{ij})$ and m is a maximal instantiation for $\mathcal{N}$. ⊣

From all this, two significant theorems follow:

Theorem 1 *Let $\mathcal{M} = (\mathcal{N}, \mathcal{S})$ be a preconvex augmented interval network. If $\mathcal{N}$ is weakly path-consistent, $\mathcal{S}$ path-consistent and if $STP(I(\mathcal{N})) = EXT(\mathcal{S})$ then there exists a maximal consistent instantiation of $\mathcal{M}$ computable in $O(n^3)$, with n the number of variables of $\mathcal{N}$.*

Theorem 2 *Let $\mathcal{M} = (\mathcal{N}, \mathcal{S}_1, \mathcal{S}_2)$ be a strongly-preconvex augmented rectangle network. If $\mathcal{N}$ is weakly path-consistent, $\mathcal{S}_1$, $\mathcal{S}_2$ path-consistent and if $STP(I(\mathcal{N})_1) = EXT(\mathcal{S}_1)$ and $STP(I(\mathcal{N})_2) = EXT(\mathcal{S}_2)$ then there exists a consistent instantiation of $\mathcal{M}$ which can be calculated in $O(n^3)$, with n the number of variables of $\mathcal{N}$.*

Proof We just prove the second theorem. $I(\mathcal{N})_1$ and $I(\mathcal{N})_2$ are path-consistent (fact 2 (c)). Consequently, from lemma 8 we deduce that $(I(\mathcal{N})_1, \mathcal{S}_1)$ and $(I(\mathcal{N})_2, \mathcal{S}_2)$ admit each one a maximal consistent instantiation. By examining the proof of lemma 8 we can see that these instantiations are calculable – with the algorithm SOLUTION-STP$^{\neq}$ for example – in $O(n^3)$ time, with n the number of variables of $\mathcal{N}$. These two consistent instantiations define a maximal consistent instantiation m of $\mathcal{N}$ (fact 2 (c)) which is a consistent instantiation of $\mathcal{M} = (\mathcal{N}, \mathcal{S}_1, \mathcal{S}_2)$. Moreover, $I(\mathcal{N})_1$ and $I(\mathcal{N})_2$ can be build in $O(n^2)$ time, we conclude that m can be find in $O(n^3)$. ⊣

Despite these theorems we cannot conclude that the consistency problem of the preconvex augmented interval and strongly-preconvex augmented rectangle networks are tractable. Before asserting that, we must characterize a polynomial method transforming a preconvex augmented interval (resp. strongly-preconvex augmented rectangle) network into an equivalent network with the adequate properties to apply theorem 1 (resp. theorem 2). We are going to prove that it is performed by the algorithm RPCM+ (see figure 3) for the strongly-preconvex augmented rectangle networks. RPCM+ can be easily modified to realize the transformation also for the preconvex augmented interval networks. By lack of place we will not write these modifications here.

Theorem 3 *Let a strongly-preconvex augmented rectangle network $\mathcal{M}$ be given as a parameter, in polynomial time the algorithm* RPCM+ *returns a consistent instantiation of $\mathcal{M}$ if $\mathcal{M}$ is consistent else it detects the inconsistency of $\mathcal{M}$.*

Proof Firstly, let us prove the soundness of RPCM+ by showing that at every step of the algorithm $\mathcal{M} = (\mathcal{N}, \mathcal{S}_1, \mathcal{S}_2)$ is equivalent to the initial network given as a parameter.
Since $WPCM$ is sound, at line 1, only atomic relations which cannot participate to consistent instantiations of $\mathcal{N}$ are removed. Moreover, $\mathcal{N}$ is still strongly-preconvex after the application of $WPCM$ (fact 1 (b)). At line 4, we remove from the $\mathcal{STP}$s $\mathcal{S}_1$ and $\mathcal{S}_2$ values not allowed by $\mathcal{O} = I(\mathcal{N})$. Since $\mathcal{N} \subseteq I(\mathcal{N})$ this operation is sound. At line 5, we know that PCM is sound. At line 8, $\mathcal{S}_1$ is path-consistent,

Algorithm RPCM+
Input: a strongly-preconvex augmented rectangle network $\mathcal{M} = (\mathcal{N}, \mathcal{S}_1, \mathcal{S}_2)$.
Output: a consistent instantiation of $\mathcal{M}$ if $\mathcal{M}$ is consistent else returns Inconsistent.
Begin
1. $\mathcal{N} := WPCM(\mathcal{N})$; (* $WPCM$ is the weak path-consistency method *)
2. If $\mathcal{N}$ contains the empty relation then **Inconsistent** ;
3. $\mathcal{O} := I(N)$;
4. $\mathcal{S}_1 := \mathcal{S}_1 \cap \mathsf{STP}(\mathcal{O}_1)$; $\mathcal{S}_2 := \mathcal{S}_2 \cap \mathsf{STP}(\mathcal{O}_2)$;
5. $\mathcal{S}_1 := PCM(\mathcal{S}_1)$; $\mathcal{S}_2 := PCM(\mathcal{S}_2)$; (* PCM is the path-consistency method *)
6. If $\mathcal{S}_1$ or $\mathcal{S}_2$ contains the empty relation then **Inconsistent** ;
7. If $\mathsf{EXT}(\mathcal{S}_1) = \mathsf{STP}(\mathcal{O}_1)$ and $\mathsf{EXT}(\mathcal{S}_2) = \mathsf{STP}(\mathcal{O}_2)$
   then return a consistent instantiation of $(\mathcal{N}, \mathcal{S}_1, \mathcal{S}_2)$ by using theorem 2.
8. $\mathcal{N} := \mathcal{N} \cap (\mathsf{IN}(\mathsf{EXT}(\mathcal{S}_1)) \times \mathsf{IN}(\mathsf{EXT}(\mathcal{S}_2)))$;
9. Goto line 1. ;

End

Figure 3: The algorithm RPCM+.

hence $\mathsf{EXT}(\mathcal{S}_1)$ is also path-consistent (lemma 4). Because of the intersection at line 4 the constraint between two variables X^- and X^+ of $\mathsf{EXT}(\mathcal{S}_1)$ (resp. $\mathsf{EXT}(\mathcal{S}_2)$), with X a variable of $\mathcal{N}_1$ (resp. $\mathcal{N}_2$), is $]0, +\infty[$. Hence $\mathsf{IN}(\mathsf{EXT}(\mathcal{S}_1))$ and $\mathsf{IN}(\mathsf{EXT}(\mathcal{S}_2))$ are defined (lemma 5). Because of the intersection at line 8, we remove from $\mathcal{N}$ the atomic relations not allowed by $\mathsf{IN}(\mathsf{EXT}(\mathcal{S}_1)) \times \mathsf{IN}(\mathsf{EXT}(\mathcal{S}_2))$. As $\mathcal{S}_1 \subseteq \mathsf{EXT}(\mathcal{S}_1)$ and $\mathcal{S}_2 \subseteq \mathsf{EXT}(\mathcal{S}_2)$ the atomic relations allowed by $\mathcal{S}_1$ and $\mathcal{S}_2$ are not removed from $\mathcal{N}$. $\mathsf{IN}(\mathsf{EXT}(\mathcal{S}_1))$ and $\mathsf{IN}(\mathsf{EXT}(\mathcal{S}_2))$ are two convex interval networks, we deduce that $\mathsf{IN}(\mathsf{EXT}(\mathcal{S}_1)) \times \mathsf{IN}(\mathsf{EXT}(\mathcal{S}_2))$ is a convex rectangle network. As a convex relation of RA is strongly-preconvex, $\mathcal{N}$ is still strongly-preconvex after the intersection (fact 1) . Hence we obtain after each step a strongly-preconvex augmented rectangle network $\mathcal{M} = (\mathcal{N}, \mathcal{S}_1, \mathcal{S}_2)$ equivalent to the initial network.

Moreover, because of $WPCM$ at line 4 and PCM at line 5, $\mathcal{N}$ is weakly path-consistent and $\mathcal{S}_1$, $\mathcal{S}_2$ are path-consistent. Finally, if $\mathsf{EXT}(\mathcal{S}_1)$ is equal to $\mathsf{STP}(I(\mathcal{N})_1)$ and $\mathsf{EXT}(\mathcal{S}_2)$ is equal to $\mathsf{STP}(I(\mathcal{N})_2)$ then we can apply theorem 2 at line 7. We conclude that RPCM+ is sound.

Now, let us prove the completeness by showing that RPCM+ stops after a finite number of steps. To do so, let us prove that $I(\mathcal{N})$ decreases in at least one atomic relation after each main loop.

$\mathcal{O}'$ will denote $I(\mathcal{N})$ at the end of the main loop. Let us show that $\mathcal{O}' \subset \mathcal{O}$. After line 7, $\mathsf{EXT}(\mathcal{S}_1) \subset \mathsf{STP}(\mathcal{O}_1)$ and $\mathsf{EXT}(\mathcal{S}_2) \subseteq \mathsf{STP}(\mathcal{O}_2)$ or $\mathsf{EXT}(\mathcal{S}_1) \subseteq \mathsf{STP}(\mathcal{O}_1)$ and $\mathsf{EXT}(\mathcal{S}_2) \subset \mathsf{STP}(\mathcal{O}_2)$. Let us suppose that the first case is satisfied, the proof of the second one is similar. $\mathsf{IN}(\mathsf{EXT}(\mathcal{S}_1)) \subset \mathsf{IN}(\mathsf{STP}(\mathcal{O}_1))$ and $\mathsf{IN}(\mathsf{EXT}(\mathcal{S}_2)) \subseteq \mathsf{IN}(\mathsf{STP}(\mathcal{O}_2))$ (lemma 6). $\mathsf{IN}(\mathsf{STP}(\mathcal{O}_1)) = \mathcal{O}_1$ and $\mathsf{IN}(\mathsf{STP}(\mathcal{O}_2)) = \mathcal{O}_2$, it follows that $\mathsf{IN}(\mathsf{EXT}(\mathcal{S}_1)) \times \mathsf{IN}(\mathsf{EXT}(\mathcal{S}_2)) \subset \mathcal{O}_1 \times \mathcal{O}_2$. Hence, $\mathsf{IN}(\mathsf{EXT}(\mathcal{S}_1)) \times \mathsf{IN}(\mathsf{EXT}(\mathcal{S}_2)) \subset \mathcal{O}$ because $\mathcal{O} = \mathcal{O}_1 \times \mathcal{O}_2$. $\mathsf{IN}(\mathsf{EXT}(\mathcal{S}_1))$ and $\mathsf{IN}(\mathsf{EXT}(\mathcal{S}_2))$ are convex, consequently $\mathsf{IN}(\mathsf{EXT}(\mathcal{S}_1)) \times \mathsf{IN}(\mathsf{EXT}(\mathcal{S}_2))$ is a convex rectangle network. Because of the intersection at line 8, $\mathcal{O}' \subseteq I(\mathsf{IN}(\mathsf{EXT}(\mathcal{S}_1)) \times \mathsf{IN}(\mathsf{EXT}(\mathcal{S}_2)))$. Since $\mathsf{IN}(\mathsf{EXT}(\mathcal{S}_1)) \times \mathsf{IN}(\mathsf{EXT}(\mathcal{S}_2))$ is convex, $I(\mathsf{IN}(\mathsf{EXT}(\mathcal{S}_1)) \times \mathsf{IN}(\mathsf{EXT}(\mathcal{S}_2))) = \mathsf{IN}(\mathsf{EXT}(\mathcal{S}_1)) \times \mathsf{IN}(\mathsf{EXT}(\mathcal{S}_2))$. Finally, we deduce that $\mathcal{O}' \subset \mathcal{O}$.

As $I(\mathcal{N})$ has at most $13^2 \times n^2$ atomic relations (with n the number of variables of $\mathcal{N}$) RPCM+ performs less than $13^2 \times n^2$ loops and is in $O(n^5)$ time since a loop takes $O(n^3)$ time. ⊣

With slightly modifications of RPCM+ we can easily obtain a polynomial algorithm allowing to find a consistent instantiation (if there is) of an augmented interval network.

7 Conclusion

We augmented the interval and the rectangle networks by $\mathcal{STP}$s. We obtained a structure more expressive allowing to deal with quantitative and qualitative constraints. We proved that the consistency problem of the strongly-preconvex augmented rectangle networks is polynomial by showing that it can be solved by the algorithm RPCM+. RPCM+ is based on the path-consistency and weak path-consistency methods. It can be slightly modify to solve the problem of the preconvex augmented interval networks in polynomial time. Currently, we are implementing this algorithm and we think to use it to solve position constraints of objects in a virtual scene [12].

References

[1] J. Allen. *Maintaining knowledge about temporal intervals.* Comm. of the ACM, Vol. 26, 832–843, 1983.

[2] P. Balbiani, J-F. Condotta and L. Fariñas del Cerro. *A model for reasoning about bidimensional temporal relations.* KR'98. Proc. of the 6th Int. Conf. Knowledge Representation and Reasoning , 124–130, Morgan Kaufmann, 1998.

[3] P. Balbiani, J-F. Condotta and L. Fariñas del Cerro. *A new tractable subclass of the rectangle algebra.* IJCAI'99. Proc. of the 16th Int. Joint Conf. on Art. Int., Stockholm, 442–447, 1999.

[4] C. A. Baykan, M. S. Fox. *Spatial synthesis by disjunctive constraint satisfaction.* Art. Int. for Engineering Design, Analysis and Manufacturing, 11, 245–262, 1997.

[5] P. van Beek. *Exact and Approximate Reasoning about Qualitative Temporal Relations*, Ph.d. Thesis of the University of Waterloo,1990.

[6] R. Dechter, I. Meiri, J. Pearl. *Temporal constraint networks.* Art. Int. 49 (1991), 61–95.

[7] A. Gerevini, M. Cristani. *On finding a solution in temporal constraint satisfaction problem.* IJCAI'97. Proc. of the 15th Int. Joint Conf. on Art. Int., 1460–1465, Vol. 2, 1997.

[8] H. Güsgen. *Spatial reasoning based on Allen's temporal logic.* Report ICSI TR89-049, International Computer Science Institute, 1989.

[9] P. Jonsson, C. Bäckström. *A Linear-Programming Approach to Temporal Reasoning.* Proc. of AAAI'96, Vol. 2, 1235–1240, 1996.

[10] H. Kautz, P. B. Ladkin. *Integrating metric and qualitative temporal reasoning.* In Proc. of AAAI'91, Araheim, CA, 1991.

[11] M. Koubarakis. *Tractable disjunctions of linear constraints.* CP'96. Proc. of the 2nd Int. Conf. on Principles and Practice of Constraint Programming, 297–301, 1996.

[12] G. Kwaiter. *Modélisation déclarative de scènes: étude et réalisation de solveurs de contraintes* . Thèse à l'université Paul Sabatier, Toulouse, 1998.

[13] G. Ligozat. *A new proof of tractability for ORD-Horn relations.* AAAI-96. Proc. of the 13th Nat. Conf. on Art. Int., Vol. 1, 395–401, 1996.

[14] I. Meiri. *Combining qualitative and quantitative constraints in temporal reasoning.* AAAI'91, 260–267, 1991.

[15] A. Mukerjee, G. Joe. *A Qualitative Model For Space.* AAAI-90. Proc. of the 8th Nat. Conf. on Art. Int., Volume two, 721–727, AAAI Press and MIT Press, 1990.

[16] B. Nebel, H.-J. Bürckert. *Reasoning about temporal relations : a maximal tractable subclass of Allen's interval algebra.* AAAI-94. Proc. of the 12th Nat. Conf. on Art. Int., Vol. 1, 356–361, 1994.

[17] K. Nökel. Temporally distributed symptoms in technical diagnosis, LNCS 517, 1991.

[18] M. Vilain, H. Kautz. *Constraint propagation algorithms for temporal reasoning.* AAAI-86. Proc. of the 5th Nat. Conf. on Art. Int., 377–382, 1986.

On the complexity of reasoning about repeating events

Robert A. Morris
RIACS/NASA Ames Research Center
MS 19-39, Moffett Field, CA 94035
morris@ptolemy.arc.nasa.gov

Paul Morris
Caelem Research/NASA Ames Research Center
MS 269-1 Moffett Field, CA 94035
pmorris@ptolemy.arc.nasa.gov

Abstract

The effective manipulation of temporal information about periodic events is often required for solving complex problems such as long range scheduling or querying temporal information. Such reasoning problems generalize reasoning problems involving single events by first, adding uncertainty about the *number* of times an event is to occur; and second, adding uncertainty about the *period* of the event, i.e., the duration between occurrences. This paper focuses on the complexity of reasoning about repeating events, expanding on the results found in an earlier paper. This paper also describes strategies for formulating and solving reasoning problems about repeating events based on a representation called Repeating Event Constraint Satisfaction Problems ($RE - CSP$).

1 Introduction

A standard representation of the time associated with an event is an interval, an ordered pair of numbers representing the start and end times of the event. Generalizing, the time associated with a *repeating event* can be represented as a set of intervals. Reasoning problems such as scheduling might involve assigning times to more than one occurrence of the same event. What distinguishes such reasoning problems from those in which each event has only one occurrence? First, there is potential uncertainty about the number of times an event is to occur. For example, instead of knowing that an event is to occur once, the problem might specify that it is to happen between three and five times. Second, there is potential uncertainty about the period between occurrences of events. For example, instead of knowing that the distance between the start times of two single-occurrence events is between three and ten minutes, it may only be known that the distance between the start times of *every* occurrence of a repeating event I and that of *some* occurrence of a repeating event J is between three and 1ten minutes. This induces uncertainty with respect to which pairs of occurrences of I and J participate in this relationship. Domains such as telescope observation scheduling [Bresina 1994] offer reasoning problems about repeating events, expressing number and period relationships in the form of constraints on the set of admissible schedules.

Previously, the computational aspects of repeating events have been discussed in the context of verifying the correctness of concurrent programs [Wolper 1983], or in the context of querying temporal relational data [Kabanza *et al.*1995]. In the AI literature, there have been studies of repeating events in the context of reasoning about calendars [Poesio and Brachman 1991], and in planning [Koomen 1991]. Most formal representations of reasoning problems of this sort are based on Temporal Logic with fix-point operators [Vardi 1988]. Such a language is too expressive for finite domain reasoning problems such as is typically found in scheduling.

The Constraint Satisfaction Problem (CSP) framework is more appropriate for solving problems with finite domains. Until recently [Morris *et al.* forthcoming], no attempt has been made to model reasoning about repeating events as a CSP. This paper expands the work of the paper just cited, wherein a formalism called Repeating event CSP ($RE - CSP$) was introduced. Specifically, this paper contains:

- A proof of a hardness result which appeared as a conjecture in the cited paper, thus expanding our knowledge of the worse-case complexity of the overall problem;

- A detailed examination of the search space associated with solving the problem of finding a consistent assignment to variables associated with number and period in an $RE - CSP$; and
- An alternative approach to verifying a solution to an $RE - CSP$ to the one offered in the cited paper. A comparison shows the potential for improvement in the time it takes to complete this step in the solution process.

The main sections of this paper are arranged as follows. To allow for a self-contained discussion, in section two the $RE - CSP$ framework is summarized. Section three leads to a proof that the uncertainty associated with knowledge about repeating events results in a class of NP-hard problems. In section four, a strategy for solving $RE - CSP$s by iterating over the set of possible ways of assigning number and period relationships is analyzed. The result of this assignment is referred to as a *concretization* of an $RE - CSP$. In section five, two distinct strategies for concretizing $RE - CSP$s are compared theoretically.

2 Representing repeating events: from instantiations to specifications

By an *instantiation* of a repeating event is meant the set of times during which the occurrences of a repeating event start or end. Given an instantiation, duration (temporal distance) information can be completely represented in the form of a set of *profiles*, of which five can be distinguished: four in terms of point combinations (end-start, start-start, start-end, and end-end), and one which displays ordering information about the intervals. For a finitely repeating event, each profile can be viewed as a matrix. Figure 1 shows an instantiation of a repeating event with three occurrences, and three profiles associated with it. Where I is an interval, let $s(I)$ and $e(I)$ stand for the start of end time of I, respectively. Each value in a profile is the difference $x(I_j) - y(I_i)$, where $x, y \in \{s, e\}$, and I_m is the m^{th} occurrence of the repeating event I. The labels under the matrices indicates the end points compared; thus, the leftmost matrix contains the values $e(I_j) - s(I_k)$, where j and k index the column and row number, respectively. The bottom profile, called the *order* profile, summarizes all the qualitative temporal relationships between pairs of occurrences of I.

Profiles have patterns that emerge from the underlying temporal structure of the repeating event, and are said to be *admissible* if they adhere to these patterns.

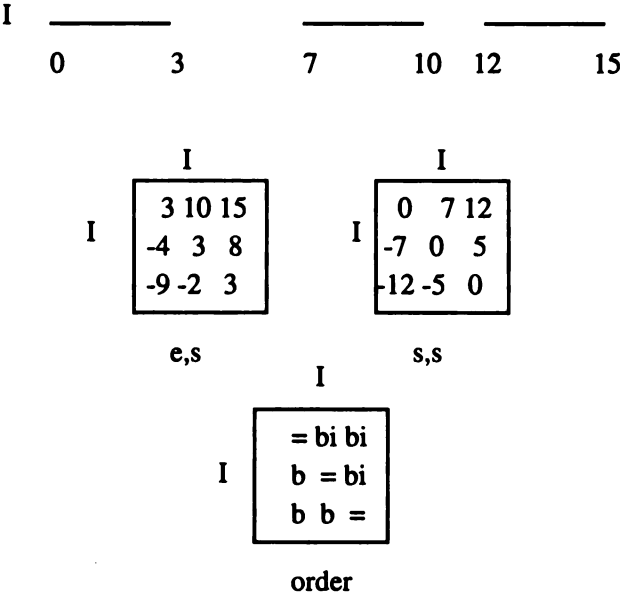


Figure 1: An instantiation of a repeating event and three profiles.

In this paper, we focus our attention on the temporal structure exhibited by repeating events all of whose instantiations are finite sets of non-overlapping intervals. Thus, in a distance profile of a finite repeating event with no overlapping occurrences, each row is a sequence of either monotonically increasing or decreasing values, as is each column. Furthermore, when the row values increase (read left-to-right), column values decrease (read top-down), and visa versa. This reflects truths about time such as if A is before B, then A is before any event that is after B. We call this the *monotonicity requirement* of profiles. Note also that every qualitative profile for a non-overlapping sequence of intervals consists of a lower-left triangle of b relations, an upper-right triangle of bi relations, and a diagonal of $=$. We call this the *tri-regional* requirement for admissible qualitative profiles.

The notion of profile can also be used to represent distance information about *pairs* of instantiations of repeating events I and J. Each value of such a profile is a distance $x(I_j) - y(J_k)$ between an end-point of an occurrence of I and one of J. We refer to this as information about the *relative profile* of two repeating events. Again, five profiles can be distinguished, and, assuming both repeating events have finite cardinality, the information can be depicted in the form of a matrix. Figure 2 illustrates relative profiles.

Relative profiles of pairs of finite, non-overlapping repeating events have monotonicity and tri-regional requirements for admissibility as well. The latter is

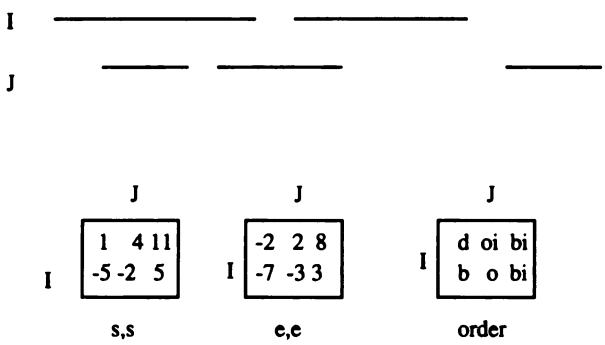


Figure 2: An instantiation of a pair of repeating event and three relative profiles.

slightly different however; the three regions of admissible profiles can be characterized as a lower-left *b* (before) region, an upper-right *bi* (after, i.e., before inverse) region, and a middle region in which any of the thirteen so-called Allen relations [Allen1983] is allowed. Thus, in Figure 2, the middle region consists of the relations during (*d*), overlapped by (*oi*), and overlaps (*o*).

In addition to the monotonicity and tri-regional requirements for admissibility of individual profiles, it is possible to characterize admissibility for *sets* of profiles in terms of adherence to constraints imposed by two *profile operations*, inverse and composition. The *inverse* $P_{I,J}^{-1}$ of a distance profile $P_{I,J}$ where $P_{I,J}[i,j] = x(J_j) - y(I_i)$ is the profile $P_{J,I}$ where $P_{J,I}[j,i] = y(I_i) - x(J_j)$ (I and J not necessarily distinct). In matrix format, the inverse of a profile is the negative transposed matrix, i.e., the one that results when the rows and columns are reversed, and the corresponding distance values negated.

Profile composition $\circ$ is defined between pairs of profiles $P_{I,J}, P_{J,K}$, where $P_{I,J}[i,j] = y(J_j) - x(I_i)$ and $P_{J,K}[l,m] = x(K_m) - z(J_l)$, $x,y,z \in \{s,e\}$. The result of $P_{I,J} \circ P_{J,K}$ is a profile $P_{I,K}$. A set P of profiles is admissible with respect to composition if, for any subset of P consisting of three profiles of the form $P_{I,J}, P_{J,K}, P_{I,K}$ $P_{I,K}[i,k] = P_{I,J}[i,j] + P_{J,K}[j,k]$, for each $j = 1 \ldots \|J\|$, where $\|J\|$ is the number of subintervals of J.

To summarize, an arbitrary set of profiles for a collection of non-overlapping repeating events is *admissible* if each distance profile in the set adheres to the monotonicity requirement for profiles, each qualitative profile is tri-regional, and the set is admissible with respect to inverse and composition.

Given a profile, it may be useful to summarize information contained in it. A summary describes a subset of the values contained in the profile. For example, a sentence like *It took all of the three group meetings fifty time units to complete*, says something about the specific distance $e(gm_3) - s(gm_1)$, a single value in a profile. Alternatively, saying something like *Each J finished three units after the completion of some I* says something about a set of distances of the form $e(J_j) - e(I_i)$. Finally, to say *none of the I's and J's overlap* is to say something about every value in a qualitative profile associated with I and J.

It follows from this characterization that it makes sense to say that a profile *satisfies* a summary. For example, the left-most profile in Figure 2 satisfies the summary *Every J starts less than five time units after some I*. In general, P satisfies s by virtue of a set of values in P. If $P_{I,J}[i,j]$ is such a value, then I_j and J_j will be said to be *correlated* with respect to s. Finally, a set S of profile summaries for a set $\mathcal{I}$ of repeating events will be called a *specification* of $\mathcal{I}$.

3 Repeating Event Constraint Satisfaction Problems

The reasoning problem of interest here can be expressed as follows: given a specification of a finite set $\mathcal{I}$ of non-overlapping, finitely repeating events, generate an admissible set of profiles for each event in $\mathcal{I}$ that satisfies each summary in the specification. This section formalizes this problem as a constraint reasoning problem.

A CSP is based on a set of variables $X = X_1, X_2, \ldots X_n$ each with a domain $Dom_i, i = 1 \ldots n$; and a set of constraints C, each of which is a relation (set of tuples) defined on a subset of X. A *solution* to a CSP is a complete assignment to elements $X_i \in X$ of values from Dom_i such that each constraint in C is satisfied. A Temporal CSP (TCSP, [Dechter *et al.*1991]) is a CSP in which the variables stand for points in time, and there are constraints for temporal distances. A Simple Temporal Problem (STP) is a TCSP in which the constraints have the form $X - Y \in [U, B]$, and says that the distance between the two events is between U and B. A standard representation for a TCSP is a network, where the nodes stand for the variables, and each edge between pairs of nodes is labeled by the interval expressing the constraint on the temporal distance between the associated variables.

A Repeating Event CSP ($RE - CSP$) is based on a set of variables representing number and period information, and constraints represented as profile summaries, as introduced above. First, the number variable $N(I)$ constrains the number of occurrences of a

repeating event I. For example, the constraint $N(I) \in [3,6] \cup [10,20]$ specifies that the number of occurrences of I should be between either three and six, or between ten and twenty. Variables for period refer to sets of profile values. First, let $E(I)$ be a variable standing for the distance between the end of the last occurrence of I and the start of the first (called the *extent* of a repeating event). The constraint $E(I) \in [30,50]$ thus states that all of I should complete after between thirty and fifty time units. Second, let Dur^I be a variable that stands for the duration of each occurrence of I. Then, the constraint $Dur^I \in [2,5]$ abbreviates the first-order formula $\forall I_i \in I\ e(I_i) - s(I_i) \in [2,5]$, and says informally that the duration of every I is between two and five time units. Third, let Gap^I be a variable that stands for the gap between successive occurrences of I. The constraint $Gap^I \in [1,2]$ abbreviates the first-order formula $\forall I_i \in I\ e(I_i) - s(I_{i-1}) \in [1,2], i > 1$, and states that the gap between successive occurrences of I is between one and two time units.

The previous variables formalize profile summaries about single repeating events. The remaining examples do the same for pairs of repeating events. Let $\forall I\ \exists J\ D^{I,J}_{s,s}$ be a variable that stands for the distance between the start of every I and the start of some J. Then, the constraint $\forall I\ \exists J\ D^{I,J}_{s,s} \in [4,10]$ abbreviates the first-order formula $\forall I_i \in I\ \exists J_j \in J\ s(J_j) - s(I_i) \in [4,10]$, and says informally that every I should start between four and ten time units before the start of some J. Similar variables standing for distances $D^{I,J}_{e,e}$, etc., can also be defined. Finally, a variable of the form $\forall I\ \forall J\ R(I,J)$. where R is a subset of the thirteen Allen relations, refers to all the values of a qualitative matrix. The constraint $\forall I\ \forall J\ \{b, bi\}(I,J)$, for example, abbreviates the expression $\forall I_i \in I\ \forall J_j \in J\ (I_i\ b\ J_j)\ \vee\ (I_i\ bi\ J_j)$, and states that there is no overlap between any I and any J.

In summary, there are qualitative, ordering constraints, and quantitative, distance constraints in a $RE - CSP$. A constraint in an $RE - CSP$ can be decomposed into three separate semantic components:

1. A pair of quantifiers defining a *mapping* between elements of I and J;
2. For each distance constraint, two additional components:
   - A pair of values in $\{s, e\}$, indicating which sets of time points from I and J are to participate in the mapping; and
   - A union of convex intervals representing the range of values of temporal distances between points.
3. For each qualitative constraint, a set of Allen relations.

Adopting previous terminology, let constraints involving single repeating events be called *unary*, and between pairs of repeating event *binary*; furthermore, by a *convex* constraint is meant one whose interval component is convex; non-convex constraints are also called *disjunctive*.

These semantic distinctions can be used to characterize a family of reasoning problems about repeating events. Here, we distinguish among four classes, and note (or prove) the complexity class of each. By an $RE - CSP\{\}$ is meant a class of reasoning problem where each specification contains only unary constraints, but allows the constraints to be disjunctive. It can easily be shown that this class of problem is NP-hard, by a simple reduction from Temporal CSPs (the proof appears in [Morris *et al.* forthcoming]). This result trivially generalizes to $RE - CSP\{b\}$, which allows, in addition, binary constraints. By contrast, problems in the class $RE - CSP\{c\}$, with specifications consisting of only convex unary constraints, can be solved in polynomial time by viewing it as a Simple Temporal Problem (STP) [Dechter *et al.*1991]. The fourth possibility, $RE - CSP\{c, b\}$ where all constraints are convex, and in which binary constraints are allowed, is the most interesting. On the one hand, it resembles a STP in having only convex constraints. The difference between a specification of an $RE - CSP\{c, b\}$ and a specification of an STP is precisely the number and period uncertainty, and therefore, given our characterization of repeating event reasoning, assigning it to a complexity class says something fundamental about problems about repeating events. That this is an NP-hard problem is demonstrated now.

Solving $RE - CSP\{c, b\}$ is $\mathcal{NP}$-hard.

We adapt the proof in [Dechter *et al.*1991], which reduces the 3-coloring problem to that of solving temporal CSPs with disjunctive constraints. Although $RE - CSP\{c, b\}$ allows only convex intervals, there is nevertheless a disjunctive character to the problem arising from the binary constraints. We use this to simulate just enough of the disjunctive constraints so that the reduction of [Dechter *et al.*1991] goes through.

The reduction of the coloring problem requires that two types of disjunctive constraints be represented. The first is a unary constraint on time points restricting them to a disjunct of intervals $\{[1,1],[2,2],[3,3]\}$. That is, the time points are restricted to values

$\{1,2,3\}$, which are interpreted as representing three different colors. The second constraint is a binary one restricting the distance between two time points to the disjunct $\{[-2,-2],[-1,-1],[1,1],[2,2]\}$. This is used to prevent adjacent nodes from having the same color. (The possibility of the two time points having a $[0,0]$ distance, i.e., being equal, is ruled out.)

As usual, it is convenient to introduce a fixed origin or zero node, and replace the unary constraints with equivalent binary ones connected from the origin. Thus, our task is to represent a network Γ of time points and the two types of binary constraints, $\{[1,1],[2,2],[3,3]\}$ and $\{[-2,-2],[-1,-1],[1,1],[2,2]\}$, within an $RE - CSP\{c,b\}$. (To avoid confusion between binary constraints on time points with binary constraints between repreating events of a $RE - CSP$, we refer to the latter henceforth in the proof as $\forall\exists$ constraints.)

We represent the time points of Γ by degenerated elements of $RE - CSP\{c,b\}$. In particular, a time point A corresponds to a "repeating event" A' that has only one occurrence of zero duration. We represent a constraint between A and B by introducing an intermediate repeating event J together with $\forall\exists$ constraint between A' and J, and between B' and J, respectively. Note that the $\forall$ quantifications are trivial over A' and B' since each has only a single sub-interval. Thus, each $\forall\exists$ constraint reduces to a simple disjunction over the sub-intervals of J. It remains to show what specific constraints on J and $\forall\exists$ constraints are needed for the two types of binary constraints discussed above.

First consider the $\{[1,1],[2,2],[3,3]\}$ constraints. In this case, we use an intermediate repeating event J that is partially degenerate; it consists of two sub-intervals, each of zero duration, with a single gap between them. The unary constraint $Gap^J \in [1,1]$ can be used to fix the gap size at $[1,1]$. We also add constraints $\forall A'\ \exists J\ D^{A',J}_{s,s} \in [1,1]$ and $\forall B'\ \exists J\ D^{B',J}_{s,s} \in [-1,-1]$. This arrangement is illustrated in figure 3. Since the Js are of zero duration, they are represented as points, designated $J1$ and $J2$. The disjunctions arising from the two constraints involve choices between $A' \rightarrow J1$ and $A' \rightarrow J2$ links on the one hand, and between $B' \rightarrow J1$ and $B' \rightarrow J2$ links on the other. Observe that the possible combinations of selections give rise to paths from A' to B' with distance bounds given by the disjunct $\{[1,1],[2,2],[3,3]\}$. For example, the selections indicated with solid lines correspond to the path $A' \rightarrow J2 \rightarrow J1 \rightarrow B'$, which produces a distance $1-1-(-1) = 1$. Each of the paths $A' \rightarrow J1 \rightarrow B'$ and $A' \rightarrow J2 \rightarrow B'$ give a distance of two. The remaining path $A' \rightarrow J1 \rightarrow J2 \rightarrow B'$ has a distance of three.

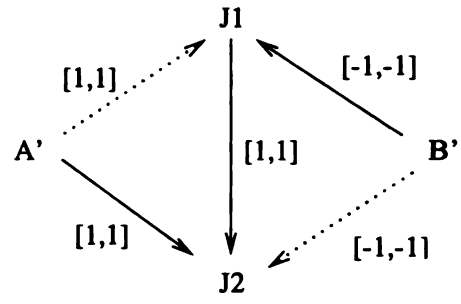


Figure 3: Representing $\{[1,1],[2,2],[3,3]\}$ constraint.

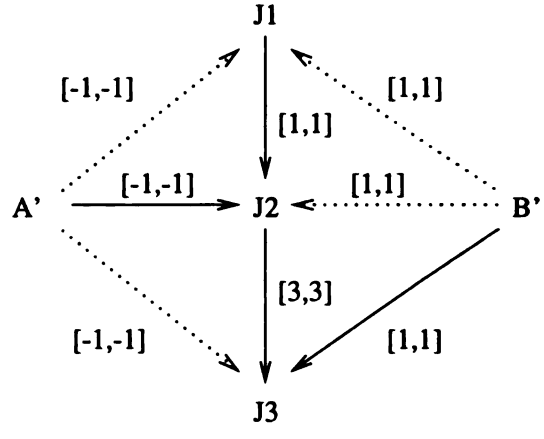


Figure 4: Representing $\{[-2,-2],[-1,-1],[1,1],[2,2]\}$ constraint.

(Note that traversing an arc in the reverse direction adds the negation of the arc's value.)

Next we tackle the more complex case of the $\{[-2,-2],[-1,-1],[1,1],[2,2]\}$ constraints. This requires an intermediate repeating event J that has three sub-intervals of zero duration and two gaps. The gap sizes are $[1,1]$ and $[3,3]$. Two variables, $\forall A'\ \exists J\ D^{A',J}_{s,s}$ and $\forall B\ \exists J\ D^{B',J}_{s,s}$ are introduced, whose values are constrained to the intervals $[-1,-1]$ and $[1,1]$, respectively. This overall arrangement is depicted in Figure 4. Now the possible paths from A' to B' have distance values within the set $\{-6,-5,-3,-2,-1,+1,+2\}$. For example, the selections indicated with solid lines correspond to the path $A' \rightarrow J2 \rightarrow J3 \rightarrow B'$, which has distance $-1+3-1 = +1$. Alternative selections produce the other distance values. Notice that the unwanted $\{-6,-5,-3\}$ possibilities are ruled out since A' and B' are already restricted to $\{1,2,3\}$ values, so their difference is always greater than or equal to -2.

It is now easy to see that a given three-coloring prob-

lem has a solution if and only if the network constructed according to the above mapping is consistent. Thus, the consistency problem for $RE-CSP\{c,b\}$ is $\mathcal{NP}$-hard. □

4 Determining consistency of specifications

The previous result justifies the examination of approaches to solving $RE-CSP$s that are provably exponential in their worst case behavior, and also for looking at approximate techniques for solving them. The remainder of this paper examines the size of the space searched by a method for determining the consistency of $RE-CSP$ specifications based on the concept of *concretization*, and also compares different structures for representing concretizations.

The notion of *concretization* was introduced in [Morris *et al.* forthcoming]. Intuitively, it is the result of transforming an $RE-CSP$ specification by assigning number to all number variables and establishing correlations between pairs of sub-intervals involved in a binary $RE-CSP\{c,b\}$ relation. Formally, for binary relations between I and J of the form $\forall I\ \exists J\mathcal{R}$, a correlation is a total mapping of indices of subintervals from I into indices of subintervals of J. Thus, correlations assume that the cardinality of I and J have been established. For example,

$$S = \{N(I) \in [1,5]; N(J) \in [3,6];$$
$$Dur^I \in [1,2];$$
$$\forall\ I \exists J D^{I,J}_{s,s} \in [2,4]\}$$

is a simple $RE-CSP\{c,b\}$ specification. Given the assignment $N(I)=4; N(J)=5$, and the relation in S between I and J, a correlation $cor_{I\to J}$ is a set of pairs of indices into sub-intervals of I and J. One such correlation can be written

$$cor_{I\to J}(1) = 1;$$
$$cor_{I\to J}(2) = 1;$$
$$cor_{I\to J}(3) = 2;$$
$$cor_{I\to J}(4) = 5.$$

The specified binary relation is transformed, given the number and correlation assignment, into the conjunction

$$s(J_1) - s(I_1) \in [2,4]\wedge$$
$$s(J_1) - s(I_2) \in [2,4]\wedge$$
$$s(J_2) - s(I_3) \in [2,4]\wedge$$
$$s(J_5) - s(I_4) \in [2,4].$$

A *concretization* of a $RE-CSP$ specification S is a description of the result of this transformation, for all number and relational constraints in S. For example, the concretization (in predicate calculus notation) for the current example is

$$N(I) = 4 \wedge N(J) = 5\wedge$$
$$s(I_1) - s(J_1) \in [2,4]\wedge$$
$$s(I_2) - s(J_1) \in [2,4] \wedge \ldots \wedge$$
$$s(I_4) - s(J_5) \in [2,4].$$

Below, other representations for concretization are discussed. Viewing a concretization C as a conjunctive formula, C is *consistent* if there is an assignment to each variable $x(I_k)$ that appears in C that makes C true.

By a *trigger* is meant the set of number assignments and correlations that produced a given concretization. For ease in the exposition that follows, a trigger is viewed as a vector (ordered set) of values. This vector is based on an ordering of the intervals in a set $\mathcal{I}$ of repeating events. For example, the tuple notation $\langle 4,5,cor_{I\to J}\rangle$ will be used represent a trigger for the example above. We write $S_T = C$ to describe the result of applying a trigger T to a specification S. Since each $RE-CSP$ specification induces a set of triggers, there is a one-to-many relationship between specifications and concretizations. Since not all resulting concretizations are consistent, the search problem arises of finding a trigger (or all triggers) that produces a consistent concretization.

The search space of this problem is the set of all possible triggers. This size of this set is potentially large. To see this, let $\mathcal{I} = \{I_1,\ldots I_N\}$ be a set of repeating events, and S a specification for $\mathcal{I}$. Let $\{N(I_j) \in [L_j,U_j]\}$ be the set of number constraints in S, $I_j \in \mathcal{I}$, and let $W_{I_j} = (U_{I_j} - L_{I_j}) + 1$. Assume that, for each pair $I_p, I_q \in \mathcal{I}$, there are pairs of constraints in S of the form $\forall I_p\ \exists I_q\ D^{I_p,I_q}_{x,y} \in [L,U]$, $\forall I_q\ \exists I_p\ D^{I_q,I_p}_{z,w} \in [L',U']$ in S. (For simplicity, we assume that there is only one instance of this binary relation in S, rather than the possible 4, one for each pair of endpoints). For each such relation, let $COR_{I_p\to I_q}[m,n]$ be the set of all ways of correlating m occurrences of I_p into n occurrences of I_q. Let $cor_{I_p\to I_q}[m,n] \in COR_{I_p\to I_q}[m,n]$. A trigger can then be viewed as the vector

$$\langle n_1, n_2, \ldots n_N, cor_{I_1\to I_2}[n_1,n_2], \ldots, cor_{I_1\to I_N}[n_1,n_N],$$
$$cor_{I_2\to I_1}[n_2,n_1], \ldots cor_{I_2\to I_3}[n_2,n_3], \ldots,$$

$$cor_{I_2 \to I_N}[n_2, n_N], \ldots cor_{I_N \to I_1}[n_N, n_1], \ldots$$

$$cor_{I_N \to I_{N-1}}[n_N, n_{N-1}]\rangle,$$

i.e., a set of number values for each $I \in \mathcal{I}$, and a set of mappings between distinct pairs $I_p, I_q \in \mathcal{I}$ involved in some binary relation. We isolate these two components as the vector $n = \langle n_1, n_2, \ldots n_N \rangle$, and the set $\mathcal{C}$, and abbreviate the entire vector as $\langle n, \mathcal{C}_n \rangle$. There are s^t ways of mapping t objects into s, and there are $O(N^2)$ correlation mappings per vector. Let $\rho_r(n)$ denote the rth element of a vector n. Each complete number assignment $n = \langle n_1, n_2, \ldots n_N \rangle$ induces up to $\Pi\ \rho_r(n)^{\rho_s(n)}, 1 \leq r, s, \leq N, r \neq s$ possible triggers. The set of all triggers is thus

$$\mathcal{T} = \bigcup_{n \in W} \langle n, \mathcal{C}_n \rangle$$

where W is the Cartesian product $W_{I_1} \times \ldots \times W_{I_N}$. Let $n^1, \ldots, n^{\|W\|}$ be an enumeration of all the members of W. The upper bound of the size of $\mathcal{T}$ is thus

$$\Sigma_{i=1}^{\|W\|}\ \Pi_{1 \leq r,s, \leq N}^{r \neq s} \rho_r(n^i)^{\rho_s(n^i)};$$

i.e., the sum, over all number assignments $n^i \in W$, of the products of all the ways to map $\rho_s(n^i)$ elements into $\rho_r(n^i)$ elements.

The size of the search space of triggers is thus exponential in the number of times a repeating event can occur. In practice, there are characteristics of the specification S that keep the search problem manageable. First, the problem might be such that N, the number of sets of repeating events in the specification, is small; this reduces the number of terms in the above product. Second, the search is reduced if the number of binary constraints in S is small, since correlations are established only for pairs of repeating events participating in a binary relation. Third, the search is reduced if tighter restrictions on the mapping relations are imposed, such as that the mapping be a bijection. Nonetheless, it may be necessary, to make a reasoning system based on concretizations effective, to apply heuristics for selecting triggers, or to apply filtering methods to find minimal specifications. The latter are discussed in [Morris *et al.* forthcoming] and the former is part of future research.

5 Consistency based on concretization

The skeletal form of an algorithm for determining the consistency of an $RE - CSP\{c, b\}$ is the following, where $\mathcal{T}$ is the set of all triggers of S.

input : a specification S of an $RE - CSP\{c, b\}$;
output : a consistent concretization C if one exists.
begin
 for each $T \in \mathcal{T}$
  $C := S_T$;
  if *consistent*(C) **then return** C;
 return fail
end

This abstract algorithm can be refined in numerous ways. For example, as discussed in [Morris *et al.* forthcoming], a trigger for a specification can be incrementally developed, as follows. At each decision point, a number variable $N(I)$ is selected for value assignment. Then, correlations are established between I and any repeating event J whose number variable $N(J)$ has been already assigned a value, and for which there is a $\forall\exists$ constraint between I and J. If the result is a consistent concretization, the process is repeated; otherwise, backtracking on either the number assignment or the correlations is performed.

The discussion in the previous section has provided an estimate of the upper bound of the number of iterations in the for-loop of the algorithm above. The remainder of the processing cost is in building a concretization and testing for consistency. The former, we assume, can be performed in time linear in the size of S. The latter can be performed in polynomial time by viewing the concretization of a $RE - CSP\{c, b\}$ as an STP, as demonstrated in [Morris *et al.* forthcoming]. The concretization into a Simple Temporal Network, for the specification S found in the previous section, and using the example trigger discussed there, is found in Figure 5. In the figure, there are four pairs of nodes representing start and end points of I, and five for J. There are also labeled arcs between the end points and the start points of J, concretizing the constraint on duration found in the specification. Finally, there are labeled arcs between start points of J and those of I, in accordance with the mapping $cor_{I \to J}$ found in the trigger. The intervals on the edges are those found in the specification for the corresponding constraint.

Concretizations using STPs are "flat" in the sense of eliminating the distinction between intervals that are part of the same repeating event and those that are not. This raises the question of whether STNs yield the most efficient concretizations of $RE - CSP$s. An alternative representation of a concretization is based on the notion of a "partial profile". A partial profile is simply a profile that consists of interval values. Partial profiles are labels of edges that connect nodes

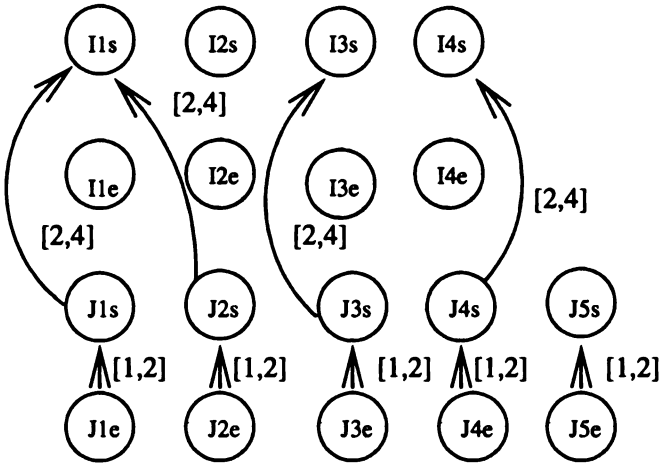


Figure 5: Concretization of a specification into a STN.

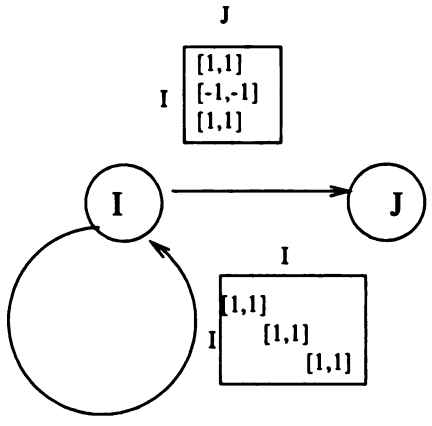


Figure 6: Concretization of inconsistent specification into a CTN.

of a network we call a Clustered Temporal Networks (CTN). Each node in a CTN represents a single repeating event. Triggers define the dimension of each partial profile, and constrain a subset of profile values, based on the correlations established in the triggers. Inverse and composition can be defined on partial profiles, generalizing these operations as defined above for complete profiles.

To illustrate a CTN, consider the following specification:

> I is a non-overlapping repeating event with three occurrences. J has one occurrence. All of the durations of the Is and J is one time unit, and there is a one time unit gap between successive occurrences of the Is. The start of J is one time unit after the start of the first I, one time unit before the start of the second I, and one time unit after the start of the third I.

This specification derives from a class of $RE - CSP$ in which there are, in addition to the variables introduced earlier, other variables which stand for specific profile elements; e.g., $D^{I,J}_{s,s}[1,2]$ stands for the distance $s(J_2) - s(I_1)$. This specification is inconsistent. To detect its inconsistency in a STN concretization, the Bellman-Ford algorithm is applied, wherein the inconsistency is determined in $O(n^3)$ steps. By contrast, Figure 6 displays the concretization of the specification into a CTN. There are two nodes in the figure, representing the 2 repeating events. Two edges are labeled by partial profiles, a 3×3 matrix representing duration constraints for occurrences of I (the diagonal), and the other a 3×1 matrix representing the binary constraint between the Is and J. Missing profile values are assumed to have the value $[-\infty, \infty]$.

Detecting inconsistency in CTNs can be performed by exploiting the admissibility requirements for profiles discussed earlier. In this example, clearly the monotonicity requirement is violated for the partial profile on the edge between I and J. For partial profiles, this requirement can be roughly stated as follows: there must exist a complete profile (i.e., ones with atomic values) selected from the intervals in the partial profile, which satisfies the requirement. Since the only solution for the profile in question is one in which the values decrease, then increase, the monotonicity requirement is violated. To check for violation of the monotonicity and tri-regional admissibility requirements, $O(N^2M^2)$ checks are made, where N is the number of nodes of the CTN, and M is the largest number of occurrences of any repeating event. This compares with $O(n^3)$ checks on a STP, where n is the number of *start or end points* of all the occurrences of any repeating event. Comparing these worst-case estimates shows that the ability of CTNs to outperform STPs in practice depends on the ability to "cluster" the reasoning problem into one involving a small number of repeating events. In this case, N, the size of the CTN, will be small, and some efficiency in determining consistency is expected.

Checking for violations of the monotonicity or tri-regional requirements for admissibility is analogous to performing arc consistency in constraint networks, insofar as only paths of length one are examined. A CTN that adheres to monotonicity requirements is not necessarily a network containing admissible profiles; the operations of composition and converse must also be preserved. To make a single computation of $P_1 \circ P_2$, a total of $O(M^3)$ comparisons must be made; thus, an

entire CTN is examined in $O(N^2M^3)$ time. Again, if the problem exhibits sufficient clustering, in practice this operation might be performed efficiently.

The thoughts just adumbrated are speculative and preliminary. They suggest that an alternative to the STP representations of concretizations for $RE-CSP$s might be justified in practice. Future work will attempt to justify these claims empirically.

6 Summary

This paper has isolated the fundamental aspects of temporal reasoning problems involving repeating events. It has also demonstrated that these aspects are sufficiently complex to render all but specialized variants of the problem NP-hard. This paper has also examined the space associated with solving number and period aspects of the reasoning problem, encapsulated in the notion of a trigger. Finally, it was claimed that reasoning about repeating events may justify the introduction of representations that exploit the internal structure of the repeating events. This paper contained a sketch of an alternative representation based on Clustered Temporal Networks.

The main theoretical result established here justifies an approach to solving $RE-CSP$s based on the method of concretization, which is worst case exponential in its performance. Current work consists of developing a system for solving $RE-CSP$s, and studying heuristics for focusing the search of the space of triggers.

References

[Allen1983] Allen, J., 1983. Maintaining Knowledge About Temporal Intervals. In *Readings in Knowledge Representation*, Brachman, R., and Levesque, H., pages 510-521, Morgan Kaufmann Publishers.

[Bresina 1994] Bresina, J.L., 1994. Telescope Loading: A Problem Reduction Approach. In *Proceedings of the Third International Symposium on Artificial Intelligence, Robotics, and Automation for Space.* Pasadena, CA, Jet Propulsion Labs.

[Dechter *et al.*1991] Dechter, R., Meiri, I., and Pearl, J., 1991 Temporal Constraint Networks. *Artificial Intelligence*, 49:61-95.

[Kabanza *et al.*1995] Kabanza, F., Stevette, J-M., and Wolper, P., 1995. Handling Infinite Temporal Data. *Journal of Computer and System Sciences*, 51, 3-17.

[Koomen 1991] Koomen, J., 1991. Reasoning about Recurrence. *International Journal of Intelligent Systems* 6:461-496.

[Morris *et al.* forthcoming] Morris, R., and Khatib, L., 2000. Constraint Reasoning about Repeating Events: Satisfaction and Optimization Forthcoming in *Computational Intelligence*, 16(2).

[Poesio and Brachman 1991] Poesio, M. and Brachman, R., 1991. Metric constraints for maintaining appointments: dates and repeated activities. In *Proceedings of AAAI-91*, pp. 253-259.

[Vardi 1988] Vardi, M., 1988 A Temporal Fixpoint Calculus. *Proceedings of ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages* 21 . San Diego, CA.

[Wolper 1983] Wolper, P., 1983 Temporal Logic Can be More Expressive. *Information and Control*, 56, 72-99.

Knowledge Engineering

Knowledge Patterns

Peter Clark, John Thompson
Knowledge Systems
Boeing Mathematics and Computing Technology
MS 7L66, PO Box 3707, Seattle, WA 98124
{peter.e.clark,john.a.thompson}@boeing.com

Bruce Porter
Computer Science Dept.
University of Texas
Austin, TX 78712
porter@cs.utexas.edu

Abstract

When building a knowledge base, one frequently repeats similar versions of general theories in multiple, more specific theories. For example, when building the Botany Knowledge Base[Porter et al., 1988], we embedded a theory of *production* in representations of *photosynthesis, mitosis, growth,* and many other botanical processes. Typically, a general theory is incorporated into more specific ones by an inheritance mechanism. However, this works poorly in two situations: when the general theory applies to a specific theory in more than one way, and when only a selected portion of the general theory is applicable.

We address this problem with a knowledge engineering technique based on the explicit representation of *knowledge patterns*, i.e., general templates denoting recurring theory schemata, and their transformation (through symbol renaming) for importing into specific theories. This technique provides considerable flexibility. A knowledge pattern may be transformed in multiple ways, and each resulting theory can be imported in whole or in part. We describe an application built using this technique, then critique its strengths and weaknesses. We conclude that this technique enables us to better modularize knowledge-bases and to reuse their general theories.

1 The Limitations of Inheritance

Consider the following fragment of a hypothetical knowledge-base about computers, expressed in Prolog[1]:

```
% Basic facts about myComputer, an instance of the class of computers:
isa(myComputer,computer). [2]
speed(myComputer,400).        /* MHz */
ram_size(myComputer, 128).    /* MB */
disk_space(myComputer,2000). /* MB */
expansion_slots(myComputer,4).

% "Available RAM space is the total RAM minus the occupied RAM."
available_ram(Computer,A) :-
    isa(Computer,computer),
    ram_size(Computer,S),
    occupied_ram(Computer,R),
    A is S - R.

% "The number of free expansion-slots is the total number of slots minus the number filled."
free_slots(Computer,N) :-
    isa(Computer,computer),
    expansion_slots(Computer,X),
    occupied_slots(Computer,O),
    N is X - O.
```

The two clauses above are syntactically different, yet they both instantiate the same general axiom, which we could explicate as:

```
FREE_SPACE(X,S) :-
    isa(X,CLASS),
    CAPACITY(X,C),
    OCCUPIED_SPACE(X,O),
    S is C - O.
```

[1] Variables start with upper-case letters and are universally quantified; ':-' denotes reverse implication (←); ',' denotes conjunction; and `is` denotes arithmetic computation.

[2] `isa(I,C)` denotes that `I` is an instance of the class `C`.

As part of a general *container* theory, this axiom relates a container's free space, capacity, and occupied space.

The clauses for `available_ram` and `free_slots` are instantiations of this axiom just when a computer is modeled as a container of data and expansion cards, respectively. However, unless this general theory of *containers* is represented explicitly, its application to the domain of computers is only implicit. Clearly, we would prefer to explicitly represent the theory, then to reuse its axioms as needed.

This is typically done with inheritance. The knowledge engineer encodes an explicit theory of *containers* at a high-level node in a taxonomy, then its axioms are automatically added to more specific theories at nodes lower in the taxonomy. One axiom in our *container* theory might be:

```
free_space(Container,F) :-
    isa(Container,container),
    capacity(Container,C),
    occupied_space(Container,O),
    F is C - O.
```

To use inheritance to import this axiom into our *computer* theory, we assert that computers are containers and that `ram_size` is a special case (a 'subslot,' in the terminology of frame systems) of the `capacity` relation:

```
% "Computers are containers."[3]
subclass_of(computer,container).

% "RAM size is a measure of capacity."
capacity(X,Y) :-
        isa(X,computer),
        ram_size(X,Y).
```

However, this becomes problematic here as there is a second notion of "computers as containers" in our original axioms, namely computers as containers of expansion cards. If we map this notion onto our *computer* theory in the same way, by adding the axiom:

```
% "Number of explansion slots is a measure
of capacity"
capacity(X,Y) :-
        isa(X,computer),
        expansion_slots(X,Y).
```

then the resulting representation captures that a computer has two capacities (memory capacity and slot capacity), but loses the constraints among their relations. Consequently, memory capacity may be used to compute the number of free expansion slots, and slot capacity may be used to compute available RAM. This illustrates how the general container theory can be "overlaid" on a computer in multiple ways, but inheritance fails to keep these overlays distinct.

This problem might be avoided in various ways. We could insist that a general theory (e.g., *container*) is applied at most once to a more specific theory (although there is no obvious, principled justification for this restriction). We would then revise our representation so that it is not a computer, but a computer's *memory*, which contains data, and similarly that a computer's *expansion slots* contain cards. While this solves the current problem, the general problem remains. For example, we may also want to model the computer's memory as a container in other senses (e.g., of transistors, files, information, or processes), which this restriction prohibits.

Another pseudo-solution is to parameterize the container theory, by adding an argument to the *container* axioms to denote the *type* of thing contained, to distinguish different applications of the *container* theory. With the changes italicized, our axioms become:

```
% "Free space for content-type T = capacity
for T - occupied T."
free_space(Container,ContentType,F) :-
  isa(Container,container),
  capacity(Container,ContentType,C),
  occupied_space(Container,ContentType,O),
  F is C - O.

% "ram_size denotes a computer's RAM ca-
pacity."
capacity(X,ram,Y) :-
  isa(X,computer),
  ram_size(X,Y).
```

Again, this solves the current problem (at the expense of parsimony), but is not a good general solution. Multiple parameters may be needed to distinguish different applications of a general theory to a more specific one. For example, we would need to add a second parameter about the container's `Dimension` (say) to distinguish physical containment (as in: "a computer contains megabytes of data") from metaphysical containment (as in: "a computer contains valuable information"). This complicates our *container* axioms further, and still other parameters may be needed.

A second limitation of inheritance is that it copies axioms (from a general theory to a more specific one)

[3] We assume a general inheritance axiom `isa(I,SuperC) :- isa(I,C), subclass_of(C,SuperC).`

in an "all or nothing" fashion. Often only a selected part of a theory should be transferred. To continue with our example, the general *container* theory may include relations for a container wall and its porosity, plus axioms involving these relations. Because the relations have no counterpart in the *computer* theory, these relations and axioms should not be transferred.

These two problems arise because inheritance is being misused, not because it is somehow "buggy." When we say "A computer is a container," we mean "A computer (or some aspect of it, such as its memory) *can be modeled as* a container." Inheritance is designed to transfer axioms through the *isa* relation, not the *can-be-modeled-as* relation. Nevertheless, knowledge engineers often conflate these relations, probably because inheritance has been the only approach available to them. This leads to endless (and needless) debates on the placement of abstract concepts in taxonomies. For example, where should *container* be placed in a taxonomy with respect to *object, substance, process* and so on? Almost anything can be thought of as a container in some way, and if we pursue this route, we are drawn into debating these modeling decisions as if they were issues of some objective reality. This was a recurrent problem in our earlier work on the Botany Knowledge-Base [Porter et al., 1988], where general theories used as models (such as *connector* and *interface*) sit uncomfortably high in the taxonomy. The same issue arises in other ontologies. For example, *product* is placed just below *individual* in Cyc [Cycorp, Inc., 1996] and *place* is just below *physical-object* in Mikrokosmos [Mahesh and Nirenburg, 1995].

2 Knowledge Patterns

Our approach for handling these situations is conceptually simple but architecturally significant because it enables us to better modularize a knowledge-base. We define a *pattern* as a first-order theory whose axioms are not part of the target knowledge-base, but can be incorporated via a renaming of their non-logical symbols.

A theory acquires its status as a pattern by the way it is used, rather than by having some intrinsic property. First, the knowledge engineer implements the pattern as an explicit, self-contained theory. For example, the *container* theory would include the axiom:

```
free_space(Container,F) :-
    isa(Container,container),
    capacity(Container,C),
    occupied_space(Container,O),
    F is C - O.
```

Second, using terminology from category theory [Pierce, 1991], the knowledge engineer defines a *morphism* for each intended application of this pattern in the target knowledge-base. A morphism is a consistent[4] mapping of the pattern's non-logical symbols, or *signature*, to terms in the knowledge-base, specifying how the pattern should be transformed. Finally, when the knowledge base is loaded, morphed copies of this pattern are imported, one for each morphism. In our example, there are two morphisms for this pattern:

```
container -> computer
capacity -> ram_size
free_space -> available_ram
occupied_space -> occupied_ram
isa -> isa
```

and

```
container -> computer
capacity -> expansion_slots
free_space -> free_slots
occupied_space -> occupied_slots
isa -> isa
```

(The reason for mapping a symbol to itself, e.g., the last line in these morphisms, is explained in the next paragraph). When these morphisms are applied, two copies of the *container* pattern are created, corresponding to the two ways, described above, in which computers are modeled as containers.

There may be symbols in the pattern that have no counterpart in the target knowledge base, such as the thickness of a *container wall* in our computer example. In this event, the knowledge engineer omits the symbols from the morphism, and the morphing procedure maps each one to a new, unique symbol (generated by Lisp's gensym function, for example). This restricts the scope of these symbols to the morphed copy of the pattern in the target knowledge base. Although the symbols are included in the imported theory, they are invisible (or more precisely, hidden) from other axioms in the knowledge base. Note that we cannot simply delete axioms that mention these symbols because other axioms in the imported theory may depend on them.[5] This selection

[4] Two examples of inconsistent mappings are: (i) mapping a symbol twice, e.g., {A->X,A->Y}, (ii) mapping a function f to g, where g's signature as specified by the mapping conflicts with g's signature as already defined in the target KB, e.g., {f->g,A->X,B->Y}, where f : A → B in the source pattern but g is already in the target and does not have signature g : X → Y.

[5] Although specific axioms may be removed if they do not contribute to assertions about symbols that are imported. A dependency analysis algorithm could, in principle, identify and remove such "dead code".

of just those predicates and functions we require corresponds to Burstall and Goguen's 'derive' operation [Burstall and Goguen, 1977] which is used to build algebraic theories from others.

Our overall approach with knowledge patterns is similar to the use of theories and morphisms in the formal specification of software (e.g., [Goguen, 1986, Srinivas and Jullig, 1995, Williamson et al., 2000]), and part of our goal is to motivate, simplify, and apply it in the context of knowledge engineering. As these authors have shown, category theory, applied to algebraic theories, provides a formal basis for this approach. To apply this to logic, Burstall and Goguen [Burstall and Goguen, 1977] show how a first-order logic theory can be understood as a many-sorted algebraic theory (comprising a set of sorts, a set of operators over those sorts, and a set of laws that those operators must satisfy) by:

- introducing truth values as a sort, and two no-argument operators (constants) `true` and `false`.
- expressing predicates as operators which produce a truth value as a result.
- Defining boolean connectives as operations (e.g., $\wedge$: boolean, boolean $\rightarrow$ boolean).
- Replacing universally quantified variables with sorts (which are implicitly universally quantified).

Drawing from [Williamson et al., 2000], we can thus define signatures, specifications (our theories or patterns), and morphisms in terms of sorted logic (rather than algebra) as follows. A *signature* consists of:

1. A set S of sort symbols
2. A triple O = <C,F,P> of operators, where C is a set of constant symbols, F is a set of function symbols, and P is a set of predicate symbols

A *specification* (corresponding to our notion of theory or pattern) consists of:

1. A signature $Sig = <S, O>$, and
2. A set Δ of axioms over Sig

A *signature morphism* (in the context of this category) is a consistent mapping from one signature to another (from sort symbols to sort symbols, and from operator symbols to operator symbols). Finally, given two specifications $<Sig_1, \Delta_1>$ and $<Sig_2, \Delta_2>$, a signature morphism M between Sig_1 and Sig_2 is a *specification morphism* between the specifications iff:

$$\forall a \in \Delta_1, (\Delta_2 \vdash M(a)) \qquad (1)$$

That is, every axiom a in Δ_1, after being translated by M, follows from Δ_2.

In our case of unsorted first-order logic, a pattern corresponds to a specification where all variables are of a single sort, and a morphism corresponds to a specification morphism. Statement (1) trivially holds because our approach deals with the special case where the resulting theory Δ_2 is by definition the translation of Δ_1 by M.

3 Using Patterns for Building a Knowledge-Base

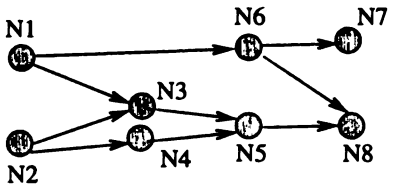
We encountered the limitations of inheritance and developed the approach of knowledge patterns while building KB-PHaSE, a prototype knowledge-based system for training astronauts to perform a space payload experiment called PHaSE (Physics of Hard Spheres Experiment). PHaSE involves projecting a laser beam through various colloidal suspensions of tiny spheres in liquids, to study the transitions among solid, liquid, and glass (not gas) states in microgravity. KB-PHaSE trains the astronaut in three ways. First, it provides a simple, interactive simulator in which the astronaut can step through the normal procedure of the experiment. Second, it introduces simulated faults to train the astronaut to recover from problems. Finally, it supports exploratory learning in which the astronaut can browse concepts in the knowledge-base and ask questions using a form-based interface. All three tasks use the underlying knowledge-base to infer: properties of the current experimental state, valid next actions, and answers to user's questions. The prototype was built as a small demonstrator, rather than for in-service use, to provide input to Boeing and NASA's Space Station Training Program. Details of KB-PHaSE are presented in [Clark et al., 1998] and the question-answering technology is described in [Clark et al., 1999].

Our interest here is how the underlying knowledge-base was assembled from component theories, rather than written from scratch. KB-PHaSE includes representations of many domain-specific objects (such as electrical circuits) and processes (such as information flow) that are derived from more general theories. For example, we can think of an electrical circuit in terms of a simple model of distribution, in which producers (a battery) distribute a product (electricity) to consumers (a light). To capture this in a reusable way, we formulated the general model of distribution as an independent, self-contained pattern, shown in the upper half of Figure 2. Then we defined a morphism that creates from it a model of electrical circuits, as shown the lower half of this Figure. Our general theory of distribution was built, in turn, by extending a general

Theory: DAG (Directed Acyclic Graph)

Synopsis

Name: dag
Summary: Core theory of directed acyclic graphs.
Uses: (none)
Used by: blockable-dag



Description: This component provides a basic axiomatization of DAGs, a fundamental structure for modeling many real-world phenomena. In a DAG, a NODE is directly linked TO and FROM zero or more other nodes [1]. A node REACHES all its downstream nodes [2], and is REACHABLE-FROM all its upstream nodes [3].
Signature: *Node*, *DAG*, *node-in*, *to*, *from*, *reaches*, *reachable-from*, *isa*.

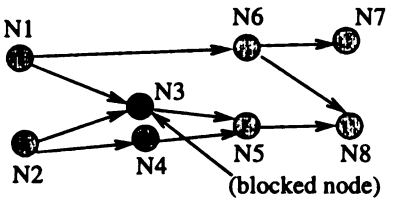
Axioms:

$\forall x, y\ to(x,y) \rightarrow isa(x, Node) \wedge isa(y, Node)$ [1]
$\forall x, y\ to(x,y) \leftrightarrow from(y,x)$
$\forall x, y\ to(x,y) \rightarrow reaches(x,y)$ [2]
$\forall x, y, z\ to(x,y) \wedge reaches(y,z) \rightarrow reaches(x,z)$
$\forall x, y\ from(x,y) \rightarrow reachable\text{-}from(x,y)$ [3]
$\forall x, y, z\ from(x,y) \wedge reachable\text{-}from(y,z) \rightarrow reachable\text{-}from(x,z)$
$\forall x, y\ isa(x, DAG) \wedge node\text{-}in(y,x) \rightarrow isa(y, Node)$

Theory: Blockable-DAG

Synopsis

Name: blockable-dag
Summary: Extension to DAG theory, in which nodes can be blocked (preventing reachability).
Uses: dag
Used by: distribution-network



Description: A NODE may be BLOCKED or UNBLOCKED [1]. A node UNBLOCKED-REACHES a downstream node if there is a path of UNBLOCKED nodes connecting the two [2].
Signature: That for dag, plus *blocked*, *unblocked*, *unblocked-directly-reaches*, *unblocked-directly-reachable-from*, *unblocked-reaches*, *unblocked-reachable-from*,
Axioms: dag theory axioms, plus:

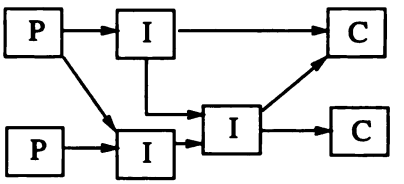
$\forall x\ isa(x, Node) \rightarrow blocked(x) \vee unblocked(x)$ [1]
$\forall x\ blocked(x) \leftrightarrow \neg unblocked(x)$
$\forall x, y\ to(x,y) \wedge \neg blocked(y) \rightarrow unblocked\text{-}directly\text{-}reaches(x,y)$
$\forall x, y\ unblocked\text{-}directly\text{-}reaches(x,y) \rightarrow unblocked\text{-}reaches(x,y)$ [2]
$\forall x, y, z\ unblocked\text{-}directly\text{-}reaches(x,y) \wedge$
$\quad unblocked\text{-}reaches(y,z) \rightarrow unblocked\text{-}reaches(x,z)$
$\forall x, y\ from(x,y) \wedge \neg blocked(y) \rightarrow unblocked\text{-}directly\text{-}reachable\text{-}from(x,y)$
$\forall x, y\ unblocked\text{-}directly\text{-}reachable\text{-}from(x,y) \rightarrow unblocked\text{-}reachable\text{-}from(x,y)$
$\forall x, y, z\ unblocked\text{-}directly\text{-}reachable\text{-}from(x,y) \wedge$
$\quad unblocked\text{-}reachable\text{-}from(y,z) \rightarrow unblocked\text{-}reachable\text{-}from(x,z)$

Figure 1: Two component theories (DAG and Blockable-DAG), used by KB-PHaSE.

Theory: Distribution Network

Synopsis

Name: `distribution-network`
Summary: Simple theory of producers, intermediaries and consumers.
Uses: `blockable-dag`
Used by: `electrical-circuit`



Description: A distribution network consists of three classes of nodes: PRODUCER, CONSUMER, and INTERMEDIARY [1], and the type of item transported is denoted by TRANSPORT-MATERIAL-TYPE (e.g., `Water`) Examples include: electrical circuits, hydraulic circuits, commuter traffic.
In this model, there is a flow of TRANSPORT-MATERIAL-TYPE from PRODUCERs to CONSUMERs via INTERMEDIARYs, providing the intermediary is not BLOCKED. A CONSUMER/INTERMEDIARY is SUPPLIED if there is at least one UNBLOCKED path to it from a SUPPLIER [2]. All elements in the network transport that network's TRANSPORT-MATERIAL-TYPE [3].
Signature: That for `blockable-dag`, plus *Producer*, *Consumer*, *Intermediary*, *Transport-Material-Type*, *supplied*, *product-type*, *consumes-type*.
Axioms: `blockable-dag` theory axioms, plus:

$\forall x\ isa(x, Producer) \rightarrow isa(x, Node)$ [1]
$\forall x\ isa(x, Consumer) \rightarrow isa(x, Node)$
$\forall x\ isa(x, Intermediary) \rightarrow isa(x, Node)$
$\forall x\ isa(x, Consumer) \wedge (\ \exists y\ isa(y, Producer) \wedge unblocked\text{-}reaches(y, x)\ ) \rightarrow supplied(x)$ [2]
$\forall x\ isa(x, Producer) \rightarrow product\text{-}type(x, Transport\text{-}Material\text{-}Type)$ [3]
$\forall x\ isa(x, Consumer) \rightarrow consumes\text{-}type(x, Transport\text{-}Material\text{-}Type)$

Theory: Electrical Circuit

Synopsis

Name: `electrical-circuit`
Summary: Top level concepts for reasoning about electrical circuits.
Uses: `distribution-network`, with morphism:

DAG	$\rightarrow$	*Electrical-Circuit*	*Producer*	$\rightarrow$	*Electrical-Power-Supply*
Node	$\rightarrow$	*Electrical-Device*	*Consumer*	$\rightarrow$	*Electrical-Appliance*
to	$\rightarrow$	*wired-to*	*Intermediary*	$\rightarrow$	*Electrical-Connector*
from	$\rightarrow$	*wired-from*	*blocked*	$\rightarrow$	*open*
supplied	$\rightarrow$	*powered*	*unblocked*	$\rightarrow$	*closed*
unblocked-reachable-from	$\rightarrow$	*circuit-between*	*consumes-type*	$\rightarrow$	*consumes-type*
Transport-Material-Type	$\rightarrow$	*Electricity*	*product-type*	$\rightarrow$	*product-type*
isa	$\rightarrow$	*isa*			

Description: In this model, an ELECTRICAL-POWER-SUPPLY provides ELECTRICITY to ELECTRICAL-APPLIANCES via ELECTRICAL-CONNECTORS. ELECTRICAL-CONNECTORS (e.g., a switch) may be OPEN (off) or CLOSED (on). An appliance is POWERED if there is an open connection from at least one power supply [1].
Signature: Morphed version of `distribution-network`, using above mapping.
Axioms: Morphed version of `distribution-network` axioms, e.g.,

$\forall x\ isa(x, Electrical\text{-}Power\text{-}Supply) \rightarrow product\text{-}type(x, Electricity)$
$\forall x\ isa(x, Electrical\text{-}Appliance) \wedge (\ \exists y\ isa(y, Electrical\text{-}Power\text{-}Supply) \wedge$
$circuit\text{-}between(y, x)\ ) \rightarrow powered(x)$ [1]

Figure 2: Component theories for distribution networks and electrical circuits, used by KB-PHaSE. The latter is defined as a morphism of the former.

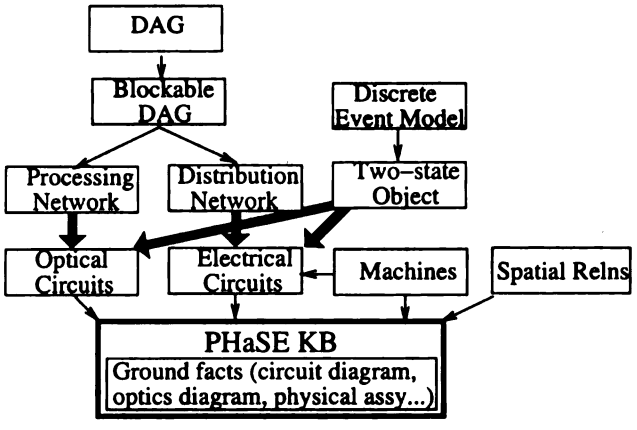


Figure 3: The component theories used in KB-PHaSE. Each box denotes a theory (set of rules) describing a phenomenon, and arcs denote inclusion relations, the thick arcs involving morphing the source.

theory of blockable directed acyclic graphs (blockable-DAGs), which in turn was built by extending a general theory of DAGs (Figure 1). The application, including these and other theories, is implemented in the frame-based language KM [Clark and Porter, 1999].

By separating these theories as modular entities, they are available for reuse. In this application, we also modeled information flow in the optical circuit (laser to camera to amplifier to disk) using a morphed pattern describing a processing network, which, in turn, was defined as an alternative extension of the basic blockable DAG theory, thus reusing this theory. Similarly, the general pattern of a "two-state object" occurs several times within KB-PHaSE (e.g., switches, lights, and open/closed covers), and this pattern was again made explicit and morphed into the knowledge base as required. These patterns and their inter-relationships are shown in Figure 3.

4 Related Work

There are several important areas of pattern-related work, differing in the type of reusable knowledge they encode and the way they encode it.

In software engineering there has been considerable work on formal methods for software specification, based on the construction and composition of theories, and using category theory (applied to algebraic specifications) as a mathematical basis (e.g., [Goguen, 1986, Srinivas and Jullig, 1995]). SpecWare is an example of a software development environment which is based on this approach and is capable of synthesizing software semi-automatically [Jullig et al., 1995]. As described in Section 2, our work can be viewed as motivating, simplifying, and applying similar ideas to the task of knowledge engineering.

Work on reusable problem-solving methods (PSMs), in particular KADS [Wielinga et al., 1992] and generic tasks [Chandrasekaren, 1986], addresses modularity and reuse in the context of procedural knowledge. PSMs are based on the observation that a task-specific method can be decomposed into more primitive – and more reusable – sub-methods, and that working with a library of such primitives may accelerate building a system and make it more understandable and maintainable. Work on PSMs shares the same general goal that we have — to identify and make explicit recurring generalizations — but it differs in two respects. First, while PSMs are (mostly) patterns of procedural inference, we have been targetting the basic domain knowledge (models) which those procedures may operate on. (Although, since logic has both a declarative and procedural interpretation, this distinction becomes blurred). Second, the mechanics of their usage differ: implementations of PSMs can be thought of as parameterized procedures, applied through instantiating their "role" parameters with domain concepts (e.g., the "hypotheses" role in a diagnosis PSM applied to medical diagnosis might be filled with disease types); in contrast, our patterns are closer to schemata than procedures, and applied instead through morphing.

Research on compositional methods for constructing knowledge bases (eg [Falkenhainer and Forbus, 1991, Clark and Porter, 1997, Noy and Hafner, 1998]) has explored factoring domain knowledge into component theories, analagous to factoring procedural knowledge into PSMs. A component theory describes relationships among a set of objects (its participants) and is applied in an analagous way to PSMs, by instantiating these participants with domain concepts. Knowledge patterns develop this idea in two ways. First, they provide further generalization, capturing the abstract structure of such theories. Second, their method of application differs (morphing, rather than axioms linking participants with domain concepts). This permits a pattern to be applied in multiple, different ways to the same object, as discussed in Section 1. Compositional modeling has also explored the automated, run-time selection of appropriate components to use [Falkenhainer and Forbus, 1991, Rickel and Porter, 1997], an important issue which we have not addressed here.

"Design patterns" in object-oriented programming (e.g., [Gamma et al., 1995]) are descriptions of common, useful organizations of objects and classes, to

help create specific object-oriented designs. They again try to capture recurring abstractions, but (in contrast to the approaches described earlier) their primary intent is as architectural guidance to the software designer, not as computational devices directly. As a result, they are (and only need be) semi-formally specified, and they do not require a method for their automatic application. ([Menzies, 1997] gives an excellent discussion of the relationship between object-oriented patterns and problem-solving methods). Another area of related work from programming languages is the use of template programming methods, where a code template is instantiated by syntactic substitution of symbols within it (e.g., Ada generics, C++ templates), corresponding to the syntactic implementation of pattern morphing, but without the associated semantics.

Work on analogical reasoning is also closely related, as it similarly seeks to use a theory (the base) to provide extra knowledge about some domain (the target), by establishing and using a mapping between the two. However, work on analogy has mainly focussed on identifying what the appropriate mappings between the base and target should be [Falkenhainer et al., 1986], a task which we have not addressed and which could be beneficial for us to explore further. In addition, an alternative way of applying our patterns would be to transform a domain-specific *problem* into the vocabulary of a pattern (and solve it there, and transform the solution back), rather than transforming the pattern into the vocabulary of the domain. In the PHaSE KB, for example, a query about the electrical circuit would be transformed to a query about a distribution network which was isomorphic to the electrical circuit, solved there, and the answer transformed back to the electrical circuit. This alternative approach is similar to (one form of) solution by analogy, in which the pattern (e.g., the distribution network) takes the role of the base, and the domain facts (e.g., the electrical circuit) the target [Falkenhainer et al., 1986]. It is also similar to the use of delegation in object-oriented programming (the target 'delegates' the problem to the base, which solves it and passes the solution back [Gamma et al., 1995, p20]). This variant approach for using patterns would allow some run-time flexibility, but would be more complex to implement and computationally more expensive at run-time.

Finally, work on microtheories and contexts (e.g., [Buvac, 1995, Blair et al., 1992]) is also related, where a microtheory (context) can be thought of as a pattern, and lifting axioms provide the mapping between predicates in the microtheory and the target KB which is to incorporate it. However, this work has typically been used to solve a different problem, namely breaking a large KB into a set of smaller, simpler (and thus more maintainable) pieces, rather than making recurring axiom patterns explicit, and it does not account for mapping the same microtheory multiple times (and in different ways) into the same target KB. Reasoning with lifting axioms can also be computationally expensive except in the simplest cases.

5 Discussion

For many representational tasks, inheritance provides a straightforward way of encoding relationships between domain-specific and abstract concepts, and the additional machinery of patterns is not necessary. Specifically, this is the case when there is a single, obvious way in which a specific concept instantiates a general one, and when all the general properties of concepts in the general theory transfer to the domain-specific ones. Psychological work on Category Structures provides support for this claim [Smith and Medin, 1981], and, computationally, inheritance is simple to understand and use.

However, as we have argued in Section 1, inheritance becomes inappropriate when a general theory can be applied in multiple ways, and when we wish to restrict how and which properties transfer to more domain-specific concepts. The pattern approach also addresses the issue of multiple theory applications through its use of morphisms (one for each application), and can also selectively transfer information, by hiding relations that do not have correlates in the target KB (Section 2). In addition, it is relatively intuitive, efficient, and easy to use.

However, our approach also has limits. First, it does not allow a system to make run-time modeling decisions, as general theories are morphed when the knowledge base is loaded. Second, it does not address the issue of *finding* relevant knowledge patterns in the first place, or deciding the appropriate boundaries of patterns (this is left to the knowledge engineer). Finally, we do not address the issue of finding the appropriate mappings between patterns and the domain; this again is left to the knowledge engineer. As mentioned earlier, this is a primary focus of research in the related field of analogical reasoning [Falkenhainer et al., 1986].

Note that patterns are not an essential prerequisite for building a knowledge-based system. In the PHaSE application, for example, we could have simply defined the PHaSE electrical circuit, implemented axioms about the behavior of electrical circuits, and answered circuit questions, all within the electrical vo-

cabulary. This would be a completely reasonable approach for a single-task system; however, to achieve reuse within a multifunctional system (such as KB-PHaSE), or between systems, it becomes preferable to extract the more general abstractions, as this paper has described. Patterns do not enable better reasoning, rather they are to help reuse.

6 Summary

In this paper, we have highlighted both the need for and difficulty of capturing and applying general theories as modular units in a KB. We have described and critiqued an approach for doing this, based on capturing those theories as "patterns" and incorporating them by morphing, and we have described an application system assembled in this way.

The significance of this approach is that it allows us to better modularize a knowledge-base and isolate general theories as self-contained units for reuse. It also allows us to control and vary the way those theories are mapped onto an application domain, and it better separates the "computational clockwork" of a general theory from the domain phenomena which it is considered to reflect. In addition, the approach is technically simple and not wedded to a particular implementation language. In the long-term, we hope this will help foster the construction of reusable theory libraries, an essential requirement for the construction of large-scale knowledge-based systems.

Acknowledgements

Thanks to Tim Menzies for valuable comments, in particular on the relationship of knowledge patterns to problem-solving methods and design patterns. Thanks also to Mike Healy, Rob Jasper, Mike Uschold, and Keith Williamson for their categorical assistance.

References

[Blair et al., 1992] Blair, P., Guha, R. V., and Pratt, W. (1992). Microtheories: An ontological engineer's guide. Tech Rept CYC-050-92, MCC, Austin, TX.

[Burstall and Goguen, 1977] Burstall, R. M. and Goguen, J. A. (1977). Putting theories together to make specifications. In *IJCAI-77*, pages 1045–1058.

[Buvac, 1995] Buvac, S., editor (1995). *Proc AAAI-95 Fall Symposium on Formalizing Context.* AAAI. http://www-formal.stanford.edu:80/bovac/95-context-symposium/.

[Chandrasekaren, 1986] Chandrasekaren, B. (1986). Generic tasks in knowledge-based reasoning: High-level building blocks for expert system design. *IEEE Expert*, pages 23–30.

[Clark and Porter, 1997] Clark, P. and Porter, B. (1997). Building concept representations from reusable components. In *AAAI-97*, pages 369–376, CA. AAAI. (www.cs.utexas.edu/users/pclark).

[Clark and Porter, 1999] Clark, P. and Porter, B. (1999). KM – the knowledge machine: Users manual. Technical report, AI Lab, Univ Texas at Austin. (http://www.cs.utexas.edu/users/mfkb/km.html).

[Clark et al., 1998] Clark, P., Thompson, J., and Dittmar, M. (1998). KB-PHaSE: A knoweldge-based training tool for a space station experiment. Technical Report SSGTECH-98-035, Boeing Applied Research and Technology, Seattle, WA. (http://www.cs.utexas.edu/users/pclark/papers).

[Clark et al., 1999] Clark, P., Thompson, J., and Porter, B. (1999). A knowledge-based approach to question-answering. In Fikes, R. and Chaudhri, V., editors, *Proc. AAAI'99 Fall Symposium on Question-Answering Systems.* AAAI. (http://www.cs.utexas.edu/users/pclark/papers).

[Cycorp, Inc., 1996] Cycorp, Inc. (1996). The cyc public ontology. (http://www.cyc.com/public.html).

[Falkenhainer and Forbus, 1991] Falkenhainer, B. and Forbus, K. (1991). Compositional modelling: Finding the right model for the job. *Artificial Intelligence*, 51:95–143.

[Falkenhainer et al., 1986] Falkenhainer, B., Forbus, K. D., and Gentner, D. (1986). The structure-mapping engine. In *AAAI-86*, pages 272–277.

[Gamma et al., 1995] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns.* Addison-Wesley.

[Goguen, 1986] Goguen, J. A. (1986). Reusing and interconnecting software components. *Computer*, pages 16–28.

[Jullig et al., 1995] Jullig, R., Srinivas, Y. V., Blaine, L., Gilham, L.-M., Goldberg, A., Green, C., McDonald, J., and Waldinger, R. (1995). Specware language manual. Technical report, Kestrel Institute. (www.kestrel.edu).

[Mahesh and Nirenburg, 1995] Mahesh, K. and Nirenburg, S. (1995). A situated ontology for practical NLP. In *Proc. IJCAI-95 Workshop on*

Basic Ontological Issues in Knowledge Sharing. (http://crl.NMSU.Edu/Research/Projects/mikro/).

[Menzies, 1997] Menzies, T. (1997). Object-oriented patterns: Lessons from expert systems. *Software – Practice and Experience*, 27(12):1457–1478. (http://www.csee.wvu.edu/~timm/).

[Noy and Hafner, 1998] Noy, N. F. and Hafner, C. D. (1998). Representing scientific experiments: Implications for ontology design and knowledge sharing. In *AAAI-98*, pages 615–622.

[Pierce, 1991] Pierce, B. (1991). *Basic Category Theory for Computer Scientists.* MIT Press.

[Porter et al., 1988] Porter, B. W., Lester, J., Murray, K., Pittman, K., Souther, A., Acker, L., and Jones, T. (1988). AI research in the context of a multifunctional knowledge base: The botany knowledge base project. Tech Report AI-88-88, Dept CS, Univ Texas at Austin.

[Rickel and Porter, 1997] Rickel, J. W. and Porter, B. W. (1997). Automated modeling of complex systems to answer prediction questions. *Artificial Intelligence.*

[Smith and Medin, 1981] Smith, E. E. and Medin, D. L. (1981). *Categories and Concepts.* Harvard Univ., Cambridge, Ma.

[Srinivas and Jullig, 1995] Srinivas, Y. V. and Jullig, R. (1995). Specware: Formal support for composing software. In *Proc. Conf. on the Mathematics of Program Construction*, Kloster Irsee, Germany. (Also Kestrel Tech Rept KES.U.94.5, http://www.kestrel.edu/HTML/publications.html).

[Wielinga et al., 1992] Wielinga, B. J., Schreiber, A. T., and Breuker, J. A. (1992). KADS: A modelling approach to knowledge engineering. *Knowledge Aquisition*, 4(1).

[Williamson et al., 2000] Williamson, K., Healy, M., and Barker, R. (2000). Reuse of knowledge at the appropriate level of abstraction. In *Proc. Sixth Int. Conf. on Software Reuse (ICSR'6).*

Knowledge Engineering by Large-Scale Knowledge Reuse — Experience from the Medical Domain

Stefan Schulz*
Medical Informatics Department
Freiburg University Hospital
Stefan-Meier-Str. 26
D-79104 Freiburg, Germany

Udo Hahn†
Text Knowledge Engineering Lab
Freiburg University
Werthmannplatz 1
D-79085 Freiburg, Germany

Abstract

We describe an ontology engineering methodology by which conceptual knowledge is extracted from an informal medical thesaurus (UMLS) and automatically converted into a formally sound description logics system. Particular attention is paid to the proper representation of partonomies which complement taxonomies as a major hierarchy-forming principle for medical knowledge. Our methodology consists of four steps: concept definitions are automatically generated from the UMLS source, integrity checking of taxonomic and partonomic hierarchies is performed by the terminological classifier, cycles and inconsistencies are eliminated, and incremental refinement of the evolving knowledge base is performed by a domain expert. We report on on-going knowledge engineering experiments in which we imported approximately 164,000 concepts and 76,000 relations into a LOOM knowledge base covering a large portion of human anatomy and pathology.

1 Introduction

Over several decades, an enormous body of medical knowledge, e.g, disease taxonomies, medical procedures, anatomical terms etc., has been assembled in a wide variety of medical terminologies, thesauri and classification systems. The conceptual structuring of a domain they allow is typically restricted to the provision of broader/narrower terms, related terms or (quasi-)synonymous terms. This is most evident in the UMLS, the *Unified Medical Language System* [9], an umbrella system which covers more than 50 medical thesauri and classifications. Its metathesaurus component contains more than 600,000 concepts which are structured in hierarchies by 134 semantic types and 54 relations between semantic types. Their semantics is shallow and entirely intuitive, which is due to the fact that their usage was primarily intended for humans. As one of the most comprehensive sources of medical terminologies it is used for various forms of clinical knowledge management, e.g., cross-mapping between different terminologies and medical information retrieval.

Given its size, evolutionary diversity and inherent heterogeneity, there is no surprise that the lack of a formal semantic foundation leads to inconsistencies, circular definitions, etc. [2]. This may not cause utterly severe problems when humans are in the loop and its use is limited to disease encoding, accountancy or document retrieval tasks. However, anticipating its use for more knowledge-intensive applications such as natural language understanding of medical narratives [5], medical decision support systems [12], etc., those shortcomings might lead to an impasse. Reuse of those rich and large resources then requires the elimination of their weaknesses.

As a consequence, formal models for dealing with medical knowledge have been proposed, using representation mechanisms based on semantic networks or description logics, such as Conceptual Graphs [21], MED [3], K-REP [8] or GRAIL [11]. Not surprisingly, however, there is a price to be paid for more expressiveness and formal rigor, *viz.* increasing modeling efforts and, hence, increasing maintenance costs. Therefore, concrete systems making full use of this rigid approach, especially those which employ high-end knowledge representation languages are usually restricted to rather small subdomains. The limited coverage then restricts their routine usage, an issue which is always highly rewarded in the medical informatics community.

* stschulz@uni-freiburg.de
† hahn@coling.uni-freiburg.de

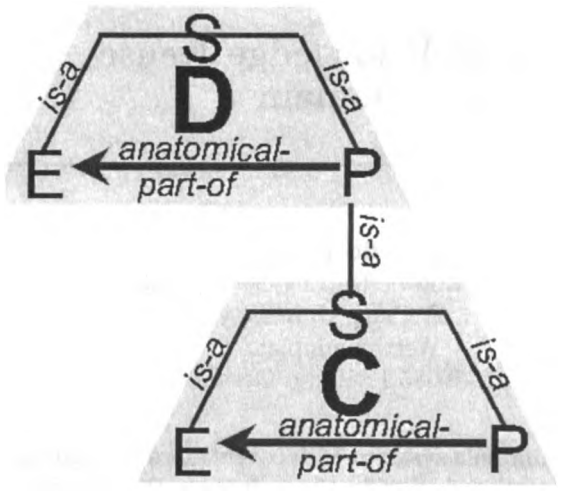


Figure 1: SEP Triplets Emulate Partitive Relations within Taxonomies

The knowledge bases developed within the framework of the above-mentioned terminological systems have all been designed from scratch – without making systematic use of the large body of knowledge contained in those medical terminologies. An intriguing approach would be to join the massive *coverage* offered by informal medical terminologies with the high level of *expressiveness* supported by formal inferencing systems, as developed in the AI knowledge representation community, in order to develop formally solid medical knowledge bases on a larger scale. This idea has already been fostered by Pisanelli et al. [10] who extracted knowledge from the UMLS semantic network as well as from parts of the metathesaurus and merged them with logic-based top-level ontologies from various sources. In a similar way, Spackman and Campbell [20] describe how the SNOMED nomenclature [4] evolves from a multi-axial coding system into a formally founded ontology. Unfortunately, the efforts made so far are entirely focused on generalization-based reasoning along *is-a* hierarchies and lack a reasonable coverage of partonomies.

2 Part-Whole Reasoning

As far as medical knowledge is concerned, two main hierarchy-building relationships can be identified, *viz.* is-a (taxonomic) and part-whole (partonomic) relations. Unlike generalization-based reasoning in concept taxonomies, no fully conclusive mechanism exists up to now for reasoning along part-whole hierarchies in description logic systems. For medical domains, however, the exclusion of part-whole reasoning is far from adequate. Anatomical knowledge, a central portion of medical knowledge, is principally organized along part-whole hierarchies. Hence, any proper medical knowledge representation has to take account of both hierarchy types [7].[1]

Various approaches to the reconstruction of part-whole reasoning within object-centered representation approaches are discussed by Artale et al. [1]. In the description logics community several language extensions have been proposed which provide special constructors for part-whole reasoning [14, 11], though at the cost of increasing computational complexity. Motivated by informal approaches sketched by Schmolze and Marks [16] and Schulz et al. [18], we recently formalized a model of part-whole reasoning [6] that does not exceed the expressiveness of the well-understood, parsimonious concept language $\mathcal{ALC}$ [15].[2]

Our proposal is centered around a particular data structure, so-called *SEP triplets*, especially designed for part-whole reasoning (cf. Figure 1). They define a characteristic pattern of IS-A hierarchies which support the emulation of inferences typical of transitive PART-OF relations. In this formalism, the relation ANATOMICAL-PART-OF describes the partitive relation between physical parts of an organism.

A triplet consists, first of all, of a composite 'structure' concept, the so-called **S-node** (e.g., INTESTINE-STRUCTURE). Each 'structure' concept subsumes both an anatomical *entity* and each of the anatomical *parts* of this entity. Unlike entities and their parts, 'structures' have no physical correlate in the real world — they constitute a representational artifact required for the formal reconstruction of systematic patterns of part-whole reasoning. The two direct subsumees of an S-node are the corresponding **E-node** ('entity') and **P-node** ('part'), e.g., INTESTINE and INTESTINE-PART, respectively. Unlike an **S-node**, these nodes refer to specific ontological objects. The E-node denotes the whole anatomical entity to be modeled, whereas the P-node is the common subsumer of any of the parts

[1] The outstanding importance of part-whole hierarchies for anatomy and, consequently, for clinical medicine has recently motivated the development of semantic networks of anatomical concepts that provide ontologically precise descriptions of partonomies though at a high granularity level [17, 13]. However, these terminological resources do not provide a formally founded methodology for part-whole reasoning.

[2] $\mathcal{ALC}$ allows for the construction of hierarchies of concepts and relations, where $\sqsubseteq$ denotes subsumption and $\doteq$ definitional equivalence. Existential ($\exists$) and universal ($\forall$) quantification, negation ($\neg$), disjunction ($\sqcap$) and conjunction ($\sqcup$) are supported. Role filler constraints (e.g., typing by C) are linked to the relation name R by a dot, $\exists R.C$.

C0005847	CHD	C0014261	part_of	MSH99	MSH99
C0005847	CHD	C0014261		CSP98	CSP98
C0005847	CHD	C0025962	isa	MSH99	MSH99
C0005847	CHD	C0026844	part_of	MSH99	MSH99
C0005847	CHD	C0026844		CSP98	CSP98
C0005847	CHD	C0034052		SNMI98	SNMI98
C0005847	CHD	C0035330	isa	MSH99	MSH99
C0005847	CHD	C0042366	part_of	MSH99	MSH99
C0005847	CHD	C0042367	part_of	MSH99	MSH99
C0005847	CHD	C0042367		SNM2	SNM2
C0005847	CHD	C0042449	isa	MSH99	MSH99

Figure 2: Semantic Relations in the UMLS Metathesaurus

of the E-node. Hence, for every P-node there exists a corresponding E-node for the role ANATOMICAL-PART-OF. Note that E-nodes and P-nodes are mutually disjoint, i.e., no anatomical concept can be ANATOMICAL-PART-OF itself.

The reconstruction of the relation ANATOMICAL-PART-OF by taxonomic reasoning proceeds as follows. Let us assume that C_E and D_E denote E-nodes, C_S and D_S denote the S-nodes that subsume C_E and D_E, respectively, and C_P and D_P denote the P-nodes related to C_E and D_E, respectively, via the role ANATOMICAL-PART-OF (cf. Figure 1). These conventions can be captured by the following terminological expressions:

$$C_E \sqsubseteq C_S \sqsubseteq D_P \sqsubseteq D_S \quad (1)$$

$$D_E \sqsubseteq D_S \quad (2)$$

The P-node is defined as follows (note the disjointness between D_E and D_P):

$$D_P \doteq D_S \sqcap \neg D_E \sqcap \exists \mathit{anatomical\text{-}part\text{-}of}.D_E \quad (3)$$

Since C_E is subsumed by D_P (1) we infer that the relation ANATOMICAL-PART-OF holds between C_E and D_E :

$$C_E \sqsubseteq \exists \mathit{anatomical\text{-}part\text{-}of}.D_E \quad (4)$$

3 Knowledge Import and Refinement

Our goal is to extract conceptual knowledge from two highly relevant subdomains of the UMLS, *viz.* anatomy and pathology, in order to construct a formally sound knowledge base using a terminological knowledge representation language. This task will be divided into four steps: (1) the automated generation of terminological expressions, (2) their submission to a terminological classifier for consistency checking, (3) the manual restitution of formal consistency in case of inconsistencies, and, finally, (4) the manual rectification and refinement of the formal representation structures. These four steps are illustrated by the workflow diagram depicted in Figure 3.

Step 1: Automated Generation of Terminological Expressions. Sources for concepts and relations were the UMLS semantic network and the *mrrel*, *mrcon* and *mrsty* tables of the 1999 release of the UMLS metathesaurus. The *mrrel* table which contains approximately 7,5 million records (cf. Figure 2) exhibits the semantic links between two UMLS CUIs (concept unique identifier),[3] the *mrcon* table contains the concept names and *mrsty* keeps the semantic type(s) assigned to each CUI. These tables, available as ASCII files, were imported into a Microsoft Access relational database and manipulated using SQL embedded in the VBA programming language. For each CUI in the *mrrel* subset its alphanumeric code was substituted by the English preferred term found in *mrcon*.

After a manual remodeling of the 135 top-level concepts and 247 relations of the UMLS semantic network, we extracted, from a total of 85,899 concepts, 38,059 anatomy and 50,087 pathology concepts from the metathesaurus. The criterion for the inclusion into one of these sets was the assignment to predefined semantic types. Also, 2,247 concepts were found to be included into both sets, anatomy and pathology. Since we wanted to keep the two subdomains strictly disjoint, we maintained these 2,247 concepts duplicated, and prefixed all concepts by ANA- or PAT- according to their respective subdomain. This can be justified

[3] As a coding convention in UMLS, any two CUIs must be connected by at least a shallow relation (in Figure 2, Child relations in the column REL are assumed between CUIs). These shallow relations may be refined in the column RELA, if a thesaurus is available which contains more precise information. Some CUIs are linked either by *part-of* or *is-a*. In any case, the source thesaurus for the relations and the CUIs involved is specified in the columns X and Y (e.g., MeSH 1999, SNOMED International 1998).

UMLS relation	number of links	Step 1 Automatic generation of Loom definitions, augmented by P-Loom language elements ;;; = *comment line*	Step 2 Submission to Loom classifier. Validation for formal consistency by Loom	Step 3 Manual restitution of formal consistency	Step 4 Manual rectification and refinement of the resulting knowledge base
Anatomy Concepts Linked to Anatomy Concepts					
sibling_of	267,218	;;; *SIB*			add negations in order to express taxonomic or partitive disjointness
child_of	59,809	;;; *CHDRN*			include related concepts into :is-primitive or :part-of clause where plausible
narrower_term	24,223	;;; *CHDRN*			
isa	9,755	:is-primitive	check for definitional cycles	remove taxonomic parent concepts	substitute primitive links by non-primitive ones where possible
location_of	4,803	;;; *LOCATION_OF*			include related concepts into :has-part clause where plausible
has_location	4,803	;;; *HAS_LOCATION*			include related concepts into :part-of clause, where plausible
has_part	4,321	:has-part			check whether this part is mandatory (under "real-anatomy" assumption)
has_conceptual_part	126				
part_of	4,321	:part-of	1. check for partonomic cycles 2. check for disjointness between E and P node	1. remove partonomic or taxonomic parent concepts 2. redefine triplet as single concept	check for plausibility and completeness
conceptual_part_of	126				
parent	59,808	;;; *PARRB*			include related concepts into :has-part clause where plausible
broader_term	24,223	;;; *PARRB*			
inverse_isa	9,755	<do nothing>			
associated_with	14				
mapped_from	2,643				
other_relation	10,908				
qualified_by	1,864				
allowed_qualifier	1,864				
mapped_to	2643				
<other named relations>	11,886	(:some x)	check for inherited constraints	remove constraints	remove or add constraints
Pathology Concepts Linked to Pathology Concepts					
sibling_of	457,642	;;; SIB			add negations in order to express taxonomic disjointness
child_of	72,426	:is-primitive	check for definitional cycles	remove parent concepts	substitute primitive links by non-primitive ones whenever possible
narrower_term	26,972				
isa	3,635				
inverse_isa	3,635	<do nothing>			
associated_with	13,902				
mapped_to	15,024				
mapped_from	15,024				
part_of	1				
has_part	1				
parent	72,426				
broader_term	28,972				
other_relation	25,796				
qualified_by	6,255				
allowed_qualifier	6,255				
<other named relations>	4,162	(:some x)	check for inherited constraints	remove constraints	remove or add constraints
Pathology Concepts Linked to Anatomy Concepts					
CUIpat = CUIana	2,247	(:some has_anatomic_correlate)			plausibility check of concept "duplication" (assignment to both domains)
<missing>		<do nothing>			add pathology-anatomy links
associated_with	2,314	(:some associated_with <patho_concept>_S)		check for consistency	render links complete, link to E-node instead of S-node when role propagation has to be disabled
has_location	9,230				
<other>		<do nothing>			

Figure 3: Workflow Diagram for the Construction of a LOOM Knowledge Base from the UMLS

```
(deftriplet HEART
   :is-primitive HOLLOW-VISCUS
   :has-part (:p-and
      ANATOMICAL-FEATURE-OF-HEART
      FIBROUS-SKELETON-OF-HEART
      WALL-OF-HEART
      CAVITY-OF-HEART
      CARDIAC-CHAMBER-NOS
      LEFT-CORONARY-SULCUS
      RIGHT-CORONARY-SULCUS
      SURFACE-OF-HEART-NOS
      LEFT-SIDE-OF-HEART
      RIGHT-SIDE-OF-HEART
      AORTIC-VALVE
      TRICUSPID-VALVE
      PULMONARY-VALVE
      MITRAL-VALVE
      HEART-VALVES-100))
```

Table 1: Generated Triplets in P-LOOM Format

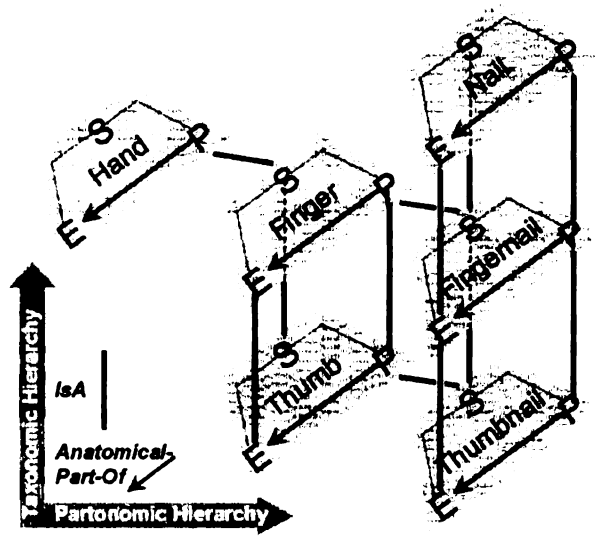


Figure 4: Example of a Mixed IS-A and PART-OF Hierarchy

by the observation that these hybrid concepts exhibit, indeed, multiple meanings. For instance, TUMOR has the meaning of a malignant disease on the one hand, and of an anatomical structure on the other hand.

As target structures for the anatomy domain we chose SEP triplets. These were expressed in the terminological language LOOM which we had previously extended by a special DEFTRIPLET macro (cf. Table 1 for an example).[4] Only UMLS *part-of*, *has-part* and *is-a* relation attributes are considered for the construction of taxonomic and partonomic hierarchies (cf. Figure 3). Hence, for each anatomy concept, one SEP triplet is created. The result is a mixed IS-A and PART-WHOLE hierarchy as depicted in Figure 4.

For the pathology subdomain, the assumption was made that *CHD* (child) and *RN* (narrower relation) relations from the UMLS stand for taxonomic links. No part-whole relations were considered, since this category does not apply to the pathology domain. Furthermore, for all anatomy concepts contained in the definitional statements of pathology concepts the 'S-node' is the default concept to which they are linked, thus enabling the propagation of roles across the part-whole hierarchy (for details, cf. [6]).

In both subdomains, shallow relations, such as the extremely frequent sibling *SIB* relation, were included as comments into the code in order to give some heuristic guidance for the subsequent manual refinement phase.

Step 2: Submission to the LOOM Classifier. The import of UMLS anatomy concepts resulted in 38,059 DEFTRIPLET expressions for anatomical concepts and and 50,087 DEFCONCEPT expressions for pathological concepts. Each DEFTRIPLET was expanded into three DEFCONCEPT (S-, E-, and P-nodes), and two DEFRELATION (ANATOMICAL-PART-OF-X, INV-ANATOMICAL-PART-OF-X) expressions, summing up to 114,177 concepts. This yielded (together with the concepts from the semantic network) a total of 240,764 definitory LOOM expressions.

From 38,059 anatomy triplets, 1219 DEFTRIPLET statements exhibited a :HAS-PART clause followed by a list of a variable number of triplets, containing more than one argument in 823 cases (average cardinality: 3.3). 4043 DEFTRIPLET statements contained a :PART-OF clause, only in 332 cases followed by more than one argument (average cardinality: 1.1). The obtained knowledge base was then submitted to the terminological classifier and checked for terminological cycles and coherence. In the anatomy subdomain, one ter-

[4]The UMLS anatomy concepts are mapped to an intermediate format, the reason being that the manual refinement of automatically generated LOOM triplets is time-consuming and error-prone due to their complex internal structure. Hence, we specified an intermediate representation language, P-LOOM, which dramatically reduces the lines of code necessary for the definition and manual inspection of triplets. P-LOOM allows to manipulate the knowledge prior to converting it to LOOM. It provides the full expressiveness of LOOM, enriched by special constructors for the encoding of the part-whole relations as well as for direct manipulation of the triplet elements whenever necessary. The main characteristics of P-LOOM are the macro DEFTRIPLET and the keywords :PART-OF and :HAS-PART. DEFTRIPLET shares the syntax of the concept-forming LOOM constructor DEFCONCEPT, augmented by the keywords :PART-OF and :HAS-PART. Both are followed by a list of SEP triplets.

	Anatomy	Pathology
Triplets	38,059	—
defconcept statements	114,177	50,087
cycles	1	355
inconsistencies	2,328	0

Table 2: Classification Results for the Import of Anatomy and Pathology Concepts

minological cycle and 2328 incoherent concepts were found, in the pathology subdomain 355 terminological cycles though not a single incoherent concept were determined (cf. Table 2).

Step 3: Manual Restitution of Consistency. The inconsistencies of the anatomy part of the knowledge base identified by the classifier could all be traced back to the simultaneous linkage of two triplets by both *is-a* and *part-of* links, an encoding that naturally raises a conflict due to the disjointness required for corresponding P-nodes and E-nodes. In most of these cases the affected parents belonged to a class of concepts that obviously cannot be appropriately modeled as SEP triplets, e.g., SUBDIVISION-OF-ASCENDING-AORTA, BODY-PART, ORGAN-PART, etc. The meaning of each of these concepts almost paraphrases that of a P-node, so that in these cases the violation of the SEP-internal disjointness condition could be accounted for by substituting the involved triplets with simple LOOM concepts, by matching them with already existing P-nodes or by disabling IS-A or PART-OF links.

In the pathology part of the knowledge base, we had expected a high number of terminological cycles, as a consequence of the decision to interpret the thesaurus-style shallow *narrower term* and *child* relations as taxonomic subsumption (IS-A). Bearing in mind the size of the knowledge base, we consider 355 cycles a tolerable amount. Those cycles were primarily due to very similar concepts, e.g., ARTERIOSCLEROSIS vs. ATHEROSCLEROSIS, AMAUROSIS vs. BINDNESS, and residual categories ("other", "NOS" = *not otherwise specified*). These were directly inherited from the source terminologies and are notoriously difficult to interpret out of their definitional context, e.g., OTHER-MALIGNANT-NEOPLASM-OF-SKIN vs. MALIGNANT-NEOPLASM-OF-SKIN-NOS. The cycles were analyzed and a negative list which consisted of 630 concept pairs was manually derived. In a subsequent extraction cycle we incorporated this list in the automated construction of the LOOM concept definitions, and given these new constraints, a fully consistent knowledge base was generated.

Step 4: Manual Rectification and Refinement of the Knowledge Base. This step – when performed for the whole knowledge base – is considerably time-consuming and requires broad and in-depth medical expertise. An analysis of random samples from both subdomains is currently being performed by a domain expert. We here state preliminary results of on-going experiments. The data we supply refer to the analysis of two random samples of each one-hundred anatomy and one-hundred pathology concepts.

From the experience we gained in the anatomy and pathology subdomains so far, the following workflow steps can be derived:

- *Checking the correctness of both the taxonomic and partitive hierarchies.* Taxonomic and partitive links are manually added or removed. Primitive subsumption is substituted by non-primitive subsumption whenever possible. This is a crucial point, because the automatically generated hierarchies contain only information about the parent concepts and necessary conditions. As an example, the automatically generated definition of DERMATITIS includes the information that it is an INFLAMMATION, and that the role HAS-LOCATION must be filled by the concept SKIN. An INFLAMMATION that HAS-LOCATION SKIN, however, cannot automatically be classified as DERMATITIS.

  Results: Taxonomic links had to be removed from 8 out of 100 sampled pathology concept definitions, but from none of the anatomy concept definitions. On the contrary, in 68 cases from this sample, anatomy concept definitions required the inclusion of anatomic or partonomic links. Often, necessary taxonomic or partonomic parents were already available, but not coded as UMLS parents or broader concepts. 7 from 100 anatomy concepts had to be considered as misclassified by the UMLS (e.g., CONGENITAL-ABSENCE-OF- TOES).

- *Check of the* :has-part *arguments assuming 'real anatomy'.* In the UMLS sources *part-of* and *has-part* relations are considered as symmetric. According to our transformation rules, the attachment of a role HAS-ANATOMICAL-PART to an E-node B_E, with its range restricted to A_E implies the existence of a concept A for the definition of a concept B. On the other hand, the classification of A_E as being subsumed by the P-node B_P, the latter being defined via the role ANATOMICAL-PART-OF restricted to B_E, implies the existence of B_E given the existence of A_E. These constraints do not always conform to 'real' anatomy, i.e., anatomical concepts that may exhibit patho-

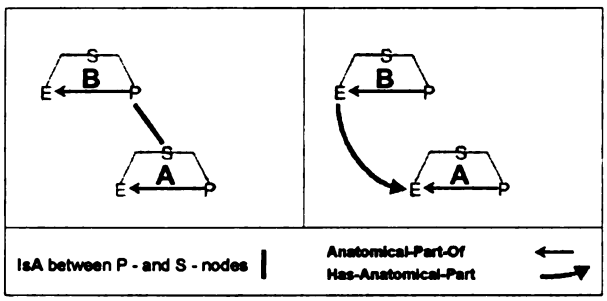


Figure 5: Patterns for Part-whole Reasoning Reconstructed Using SEP triplets: ANATOMICAL-PART-OF without HAS-ANATOMICAL-PART (Left), HAS-ANATOMICAL-PART without ANATOMICAL-PART-OF (Right)

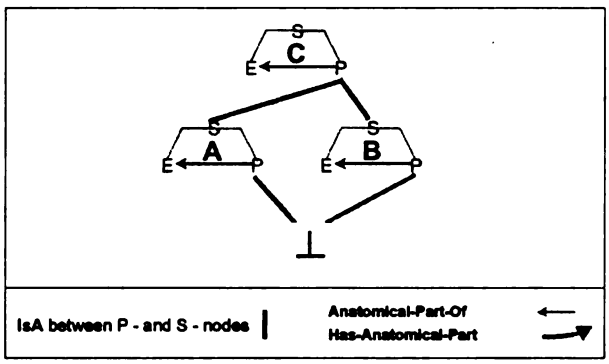


Figure 6: Spatial Disjointness between Triplet Constructs

logical modifications. Figure 5 (left) sketches a concept A that is necessarily ANATOMICAL-PART-OF a concept B, but whose existence is not required for the definition of B. This is typical of the results of surgical interventions, e.g., a large intestine without an appendix, or an oral cavity without teeth, etc.

Results: The analysis of 15 triplet definitions that exhibit automatically generated :HAS-PART clauses revealed that 34% of the concepts should be eliminated from the :HAS-PART list in order not to obviate a coherent classification of pathologically modified anatomical objects.[5] The opposite situation is also common (cf. Figure 5, right): the definition of A_E does not imply that the role ANATOMICAL-PART-OF be filled by B_E, but B_E does imply that the inverse role be filled by A_E. As an example, a LYMPH-NODE necessarily contains LYMPH-FOLLICLES , but there exist LYMPH-FOLLICLES that are not part of a LYMPH-NODE.

- *Analysis of the sibling relations and defining concepts as being disjoint.* The UMLS relation *SIB* relates concepts that share the same parent in a taxonomic or partonomic hierarchy. Pairs of sibling concepts may have common descendants or not. If not, they constitute the root of two disjoint subtrees. In a taxonomic hierarchy, this means that one concept implies the negation of the other (e.g., a benignant tumor cannot be a malignant one, *et vice versa*). In a partitive hierarchy, this can be interpreted as *spatial* disjointness, *viz.* one concept does not spatially overlap with another one.

[5] In the example of Table 1, the concepts printed in *italics*, *viz.* AORTIC-VALVE, TRICUSPID-VALVE, PULMONARY-VALVE and MITRAL-VALVE should be eliminated from the :HAS-PART list, because they may be missing in certain cases as a result of congenital malformations, inflammatory processes or surgical interventions.

In our triplet formalism this can be expressed as follows: Spatial disjointness refers to a pair of concepts A and B, whose S-nodes, A_S and B_S are subsumed by the P-node C_P. Consequently, a relation ANATOMICAL-PART-OF holds between A_E and C_E, as well as between B_E and C_E. If A and B are spatially disjoint, there does not exist any concept that is both ANATOMICAL-PART-OF A_E and ANATOMICAL-PART-OF B_E, (cf. Figure 6). As an example, ESOPHAGUS and DUODENUM are spatially disjoint, whereas STOMACH and DUODENUM are not (they share a common transition structure, called PYLORUS), such as all neighbor structures that have a surface or region in common. Spatial disjointness can be modeled in a way that the definition of the S-node of the concept A implies the negation of the S-node of the concept B [19].

Results: The large number of sibling concepts (on the average 7.3 siblings per concept in the anatomy subdomain, and 8.8 in the pathology subdomain) makes the modeling of disjointness a time-consuming task, as every pair of concepts must be analyzed. At first glance, our data indicate that conceptual disjointness can be assumed for at least two-thirds of the sibling concepts in both domains, and spatial disjointness for over three quarters in the anatomy domain.

- *Completion and modification of anatomy–pathology relations.* Surprisingly, only very few pathology concepts contained an explicit reference to a corresponding anatomy concept. These relations must, therefore, be added by a domain expert. In each case, the decision must be made whether the E-node or the S-node has to be addressed as the target concept for modification. In the first case, the propagation of roles across part-whole hierarchies is disabled. As

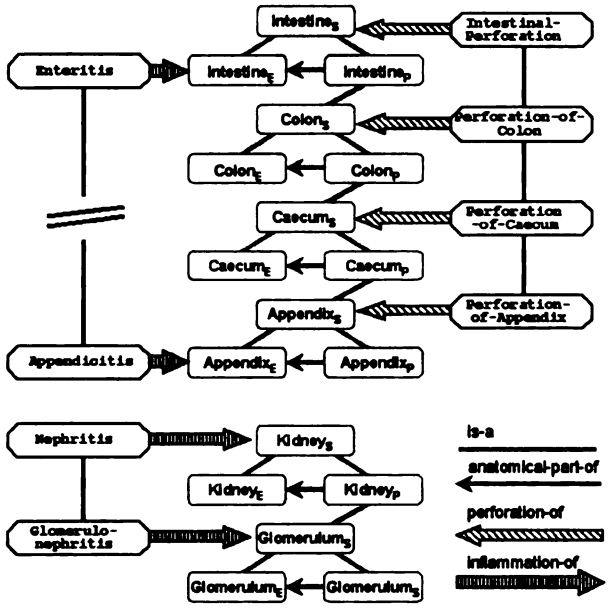


Figure 7: Linkage of a Pathology Concept Alternatively to the S-node or the E-node of an Anatomy Triplet

an example (cf. Figure 7), ENTERITIS is defined as INFLAMMATION-OF INTESTINE. The range of the relation INFLAMMATION-OF is restricted to the E-node of INTESTINE. This precludes, for instance, the classification of APPENDICITIS as ENTERITIS though the APPENDIX is related to the INTESTINE via an ANATOMICAL-PART-OF relation. In the second case, the target is the S-node of the anatomic triplet, and the propagation of roles is enabled. GLOMERULONEPHRITIS is therefore classified as NEPHRITIS (INFLAMMATION-OF KIDNEY), the GLOMERULUM being a PART-OF the KIDNEY. In the same way, PERFORATION-OF-APPENDIX is classified as INTESTINAL-PERFORATION (cf. [6] for a more comprehensive analysis of these phenomena).

Results: In an analysis of a random sample of 100 pathology concepts, only 17 were found to be linked with an anatomy concept. In 15 cases, the default linkage to the S-node was considered to be correct, in one case the linkage to the E-node was preferred. In another case, the linkage was considered false.

4 Conclusions

There is a growing demand for high-quality terminology services and their embedding in functionally advanced health information systems. Instead of developing sophisticated medical knowledge bases from scratch, we here propose a 'conservative' approach — reuse existing large-scale resources, but refine the data from these resources so that advanced representational requirements imposed by more expressive knowledge representation languages are met. The resulting knowledge bases can then be used for sophisticated applications requiring formally sound medical reasoning.

The ontology engineering methodology we have proposed in this paper does exactly this. It provides a formally solid description logics framework with a modeling extension by SEP triplets so that both taxonomic and partonomic reasoning are supported equally well. While plain automatic conversion from semi-formal to formal environments causes problems of adequacy of the emerging representation structures, the refinement methodology we propose already inherits its power from the terminological reasoning framework. In our concrete work, we found the implications of using the terminological classifier, the inference engine which computes subsumption relations, of utmost importance and of outstanding heuristic value. Hence, the knowledge refinement cycles are truly semi-automatic, fed by medical expertise on the side of the human knowledge engineer, but also driven by the reasoning system which makes explicit the consequences of (im)proper concept definitions.

We also stipulate that our knowledge engineering methodology is general in the sense that it can be applied to all those domains where, at least, shallow thesauri are available. Our focus on partonomic knowledge and reasoning is, of course, best matched by science and engineering domains (e.g., we might also consider using the INSPEC thesaurus for physics, electronic engineering and computer science).

Acknowledgements. We want to thank our colleague Martin Romacker for close cooperation and instant support. Stefan Schulz is supported by a grant from DFG (Ha 2097/5-1).

References

[1] Alessandro Artale, Enrico Franconi, Nicola Guarino, and Luca Pazzi. Part-whole relations in object-centered systems: an overview. *Data & Knowledge Engineering*, 20(3):347–383, 1996.

[2] James J. Cimino. Auditing the Unified Medical Language System with semantic methods. *Journal of the American Medical Informatics Association*, 5(1):41–45, 1998.

[3] James J. Cimino, Paul D. Clayton, George Hripsack, and Stephen B. Johnson. Knowledge-based

approaches to the maintenance of a large controlled medical terminology. *Journal of the American Medical Informatics Association*, 1(1):35–50, 1994.

[4] Roger Côté. *SNOMED International*. Northfield, IL: College of American Pathologists, 1993.

[5] Udo Hahn, Martin Romacker, and Stefan Schulz. How knowledge drives understanding: matching medical ontologies with the needs of medical language processing. *Artificial Intelligence in Medicine*, 15(1):25–51, 1999.

[6] Udo Hahn, Stefan Schulz, and Martin Romacker. Partonomic reasoning as taxonomic reasoning in medicine. In *AAAI'99/IAAI'99 – Proceedings of the 16th National Conference on Artificial Intelligence & 11th Innovative Applications of Artificial Intelligence Conference*, pages 271–276. Orlando, Florida, July 18-22, 1999. Menlo Park, CA; Cambridge, MA: AAAI Press & MIT Press, 1999.

[7] Ira J. Haimowitz, Ramesh S. Patil, and Peter Szolovits. Representing medical knowledge in a terminological language is difficult. In R. A. Greenes, editor, *SCAMC'88 – Proceedings of the 12th Annual Symposium on Computer Applications in Medical Care*, pages 101–105. New York, N.Y.: IEEE Computer Society Press, 1988.

[8] Eric Mays, Robert Weida, Robert Dionne, Meir Laker, Brian White, Chihong Liang, and Frank J. Oles. Scalable and expressive medical terminologies. In J. J. Cimino, editor, *Proceedings of the 1996 AMIA Annual Fall Symposium (formerly SCAMC). Beyond the Superhighway: Exploiting the Internet with Medical Informatics*, pages 259–263. Washington, D.C., October 26-30, 1996. Philadelphia, PA: Hanley & Belfus, 1996.

[9] Alexa T. McCray and S. J. Nelson. The representation of meaning in the UMLS. *Methods of Information in Medicine*, 34(1/2):193–201, 1995.

[10] Domenico M. Pisanelli, Aldo Gangemi, and Geri Steve. An ontological analysis of the UMLS metathesaurus. In C. G. Chute, editor, *Proceedings of the 1998 AMIA Annual Fall Symposium. A Paradigm Shift in Health Care Information Systems – Clinical Infrastructures for the 21st Century*, pages 810–814. Orlando, FL, November 7-11, 1998. Philadelphia, PA: Hanley & Belfus, 1998.

[11] Alan L. Rector, Sean Bechhofer, Carole A. Goble, Ian Horrocks, W. Anthony Nowlan, and W. Danny Solomon. The GRAIL concept modelling language for medical terminology. *Artificial Intelligence in Medicine*, 9:139–171, 1997.

[12] James A. Reggia and S. Tuhrim, editors. *Computer-Assisted Medical Decision Making*, volume 1 & 2. New York: Springer, 1985.

[13] Cornelius Rosse, Linda G. Shapiro, and James F. Brinkley. The Digital Anatomist foundational model: principles for defining and structuring its concept domain. In C. G. Chute, editor, *Proceedings of the 1998 AMIA Annual Fall Symposium. A Paradigm Shift in Health Care Information Systems – Clinical Infrastructures for the 21st Century*, pages 820–824. Orlando, FL, November 7-11, 1998. Philadelphia, PA: Hanley & Belfus, 1998.

[14] Ulrike Sattler. A concept language extended with different kinds of transitive roles. In G. Görz and S. Hölldobler, editors, *Advances in Artificial Intelligence. Proceedings 20th Annual German Conference on Artificial Intelligence – KI'96*, pages 333–345. Dresden, Germany, September 17-19, 1996. Berlin: Springer, 1996. (Lecture Notes in Artificial Intelligence, 1137).

[15] Manfred Schmidt-Schauß and Gerd Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[16] James G. Schmolze and W. S. Marks. The NIKL experience. *Computational Intelligence*, 6:48–69, 1991.

[17] Rainer Schubert and Karl Heinz Höhne. Partonomies for interactive explorable 3D-models of anatomy. In C. G. Chute, editor, *Proceedings of the 1998 AMIA Annual Fall Symposium. A Paradigm Shift in Health Care Information Systems – Clinical Infrastructures for the 21st Century.*, pages 433–437. Orlando, FL, November 7-11, 1998. Philadelphia, PA: Hanley & Belfus, 1998.

[18] Erich B. Schulz, Colin Price, and Philip J. B. Brown. Symbolic anatomic knowledge representation in the Read Codes Version 3: structure and application. *Journal of the American Medical Informatics Association*, 4(1):38–48, 1997.

[19] Stefan Schulz, Martin Romacker, Giovanni Faggioli, and Udo Hahn. From knowledge import to knowledge finishing: automatic acquisition and semi-automatic refinement of medical knowledge.

In B. Gaines, R. Kremer, and M. Musen, editors, *KAW'99 - Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management*, pages 7/8/1–7/8/12. Banff, Alberta, Canada, October 16-21, 1999.

[20] Kent A. Spackman and Keith E. Campbell. Compositional concept representation using SNOMED: towards further convergence of clinical terminologies. In C. G. Chute, editor, *Proceedings of the 1998 AMIA Annual Fall Symposium. A Paradigm Shift in Health Care Information Systems: Clinical Infrastructures for the 21st Century.*, pages 740–744. Orlando, FL, November 7-11, 1998. Philadelphia, PA: Hanley & Belfus, 1998.

[21] F. Volot, Pierre Zweigenbaum, Bruno Bachimont, M. Ben Said, Jacques Bouaud, Marius Fieschi, and Jean-François Boisvieux. Structuration and acquisition of medical knowledge: using UMLS in the Conceptual Graph formalism. In C. Safran, editor, *SCAMC'93 - Proceedings of the 17th Annual Symposium on Computer Applications in Medical Care*, pages 710–714. New York: McGraw-Hill, 1994.

A Logic Based Language for Parametric Inheritance

Hasan M. Jamil
Department of Computer Science
Mississippi State University, USA
jamil@cs.msstate.edu
www.cs.msstate.edu/~jamil/

Abstract

Though overriding as a single and default mode of inheritance is adequate for most knowledge bases, a large class of applications naturally requires several inheritance *modes* and *types*. We propose linguistic extensions of logical object-oriented languages to support multiple inheritance modes and types and give up the usual default overriding semantics in object-oriented systems. In this paper, we present a declarative semantics for a generalized parametric language for inheritance in object-oriented knowledge-bases. We specifically identify and discuss two inheritance modes, *overriding* and *inflating*, and three inheritance types, *value, code* and *null*. The parameterization we consider for our language is at the clause level. This approach generalizes to most languages similar to ours. We present a formal account of our language by giving a novel proof theory. We also hint at a model theory and a fixpoint theory and claim their equivalence. A full logical characterization of modes and types in a single framework as we present here is, to our knowledge, unprecedented. We also show that languages with default overriding semantics and the classical Horn clause language are special cases of ours. Finally, we discuss modeling modes and types as computable functions as a future extension.

1 Introduction

Perhaps the connotations were different, but inheritance has found its way into numerous systems and applications in all its different forms. Of the many possible kinds proposed, inheritance with overriding has been used and appreciated most widely. Overriding has been a difficult issue to address in logic, as it is related to negation, belief or theory revision, and non monotonic reasoning. Several approaches were appealing in modeling overriding in knowledge representation and in knowledge bases [9, 38, 11, 4, 26, 8, 35, 6, 22, 5, 31, 5, 2, 3, 29, 28, 39, 13, 14, 30, 19]. Most solutions were computationally expensive, or were heavily dependent on the user for capturing the intended semantics – i.e, inheritance was almost hand coded into the program. In the latter case, logic did very little in capturing the spirit of overriding and thus, was error prone. Currently, there are varied frameworks for computing declarative overriding. Among these, inheritance in object-oriented logic is of very recent interest in the area of object-oriented knowledge bases [34, 40, 1, 21, 12, 7, 10, 25, 19] (comprehensive surveys may be found in [9, 21]). This interest has been sparked primarily by the need to model new generation knowledge applications such as bio-informatics, GIS, CAD/CAM, office automation, web databases, time series expert systems, etc. that require complex abstractions and necessitate software engineering concepts. In these applications as well, inheritance with overriding has been the predominant choice.

Inheritance has also made significant inroads to applications in other areas including multi-level secure databases [20], data warehousing [16], data mining [15], etc. Though overriding has played a pivotal role, by itself, it was not enough for representing knowledge in these systems. In F-logic [21], monotonic, *additive* or *inflation* inheritance was needed to model structure or signature inheritance, while overriding was instrumental for capturing method inheritance. Modeling both was difficult in F-logic and a proof procedure for method inheritance with overriding was not possible. The main reason was that the logic was very general, and unlike many other languages it was made free from cumbersome syntax, hardwiring inheritance

into the programs. ORLog [19] succeeded in capturing inheritance with overriding within its proof theory by restricting F-logic and devising a special syntax, yet keeping the logic purely declarative. But ORLog differs from F-logic in a significant way. ORLog captured the notion of *code* (i.e., clause) inheritance and *late binding* (as defined in definition 3.2) which F-logic could only simulate, not model in a declarative sense. In other words, F-logic is only capable of *value* inheritance (i.e., results of computations or facts) and simulates code inheritance using the machinery of value inheritance. Yet another type of inheritance was used recently in multi-level secure knowledge bases, called the *null* inheritance, to model the so called *conservative view* of belief to remove ambiguity in classified knowledge bases [20].

In light of the preceding discussion, we envisage that a unifying and generalized language model for inheritance is imperative so that knowledge representation can become less burdened and cumbersome, and can be used for larger sets of application modeling. In a later section, we use an example to demonstrate that such extensions are natural, intuitive and interesting. In the discussion to ensue, we call overriding and inflation *modes* (of inheritance), and value, code and null *types* (of inheritance). It is our contention that incorporating modes and types makes it easier to represent knowledge in object-oriented logics with a hierarchy of classes, objects and methods. While several others are possible, in this paper we consider only two modes and three types of inheritance which we believe are sufficient for modeling most applications. An interesting example is discussed next which demonstrates the need for multiple modes and types of inheritance in a single knowledge representation application.

1.1 A Motivating Example

We highlight the necessity for multiple inheritance modes and types with a university database example presented using OR data model [18] like schema in figure 1 for clarity and simplicity, omitting several details of the model in order to be succinct and focused.

Intuitively, every object in a class has an object identity and a set of properties. Properties exhibit the "behavior" of the objects, or instances, (and also classes). The behaviors, often called the *methods*, may be defined statically as *values*, or dynamically as *codes* that compute the values corresponding to a property at run time. Behaviors defined at class objects are called *default* behaviors. Default behaviors may be inherited at a subclass lower in the hierarchy, or at an instance, unless a subclass or instance *overrides* it by re-defining the behavior. Static values defined at the class level as defaults are often called the state values in object oriented parlance.

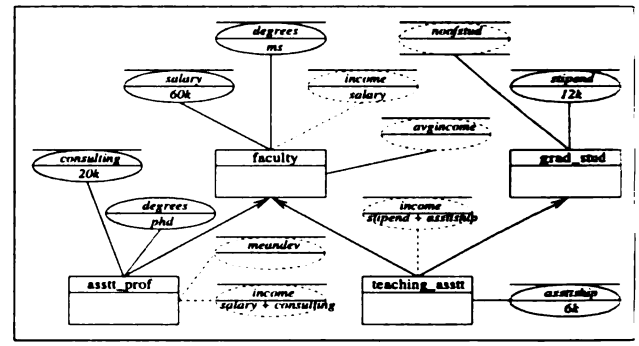


Figure 1: University database schema in extended OR notations showing inheritance modes and types.

While inheriting behaviors along the class hierarchies, a significant distinction can be identified, resulting in two very different meanings of behavioral inheritance. A property which is statically defined, alternatively called a static property, can only be statically inherited – i.e., only the value can be inherited. This form of inheritance is called *value* inheritance. But the dynamic properties, or methods, that are defined via codes, can be inherited in two distinct forms: (i) the code computed at the class where the property is defined and the *"computed value"* inherited similar in spirit to value inheritance discussed earlier; alternatively, (ii) the code as a whole inherited and computed in the context of the object of interest. This form of inheritance is termed as *code reuse*, or *code* inheritance, and is similar to the idea of *dynamic self binding*, or *late binding* in object-oriented languages such as C++. Conceptually, these two forms of inheritances are quite different and have significant influence on the semantics of a program, as will be demonstrated shortly.

Let faculty, asstt_prof, grad_stud and teaching_asstt be four class objects, where asstt_prof and teaching_asstt are subclasses of faculty. teaching_asstt is also a subclass of grad_stud. The attributes of each of the classes are shown in ellipses, with the name of the attributes shown above the horizontal line and the values shown below. For example, *degrees* is an attribute of the faculty class and has a default value *ms* for all faculty instances. Note that the *income* attribute in the class asstt_prof is a virtual attribute, or a *methods*, and is shown as a dotted ellipse in figure 1. The *income* for asstt_prof is defined to be the sum total of the *salary* and *consulting* attribute values. The thick upward arrow lines in figure 1 show the is-a relationships among class objects.

However, there are several unique properties of the underlying data model of our language. Consider the attribute *degrees* in faculty and asstt_prof. We naturally expect that the qualifications of asstt_profs is a superset of the qualifications of facultys, and hence, are additive. We call it *inflating* as opposed to overriding. In contrast, the method *income* in asstt_prof is a specialized definition of the *income* definition in faculty and hence, should be overridden. In figure 1, these two modes of inheritance are shown by two different attribute notations. The attributes or methods that override a more generalized definition have a horizontal bar on the top of the ellipse and the inflating attributes are shown without a bar. The type of a property/method, on the other hand, is illustrated as follows: a dotted line connecting the entity rectangle and the property ellipse denoting a code inheritance while a solid line represents a value inheritance. A horizontal bar under the property ellipse symbolizes a non-inheritable property. In figure 1, *noofstud* in grad_stud is such a property.

Consequently in this example, we expect that every asstt_profs will have an *income* equivalent to 80K while their *salary* has a value 60K which is inherited from faculty. Every asstt_profs will have *degrees* $\{ms, phd\}$. It should be clear that if we only have overriding as the default underlying semantics, the only way we can force *degrees* to assume the values $\{ms, phd\}$ in asstt_profs is by defining the value for the *degrees* attribute to be so. But then the problem is that if an attribute is defined via some program code, say in faculty, we are then forced to copy that piece of code from faculty to, say, asstt_prof. Duplicating codes raises the issue of code maintenance and potential inconsistency. This is a major software engineering concern which we must avoid if possible.

1.2 Goals and Contributions of this Paper

The goal of this paper is to develop a logical framework for parametric inheritance in knowledge-base systems, and to develop a query language for such knowledge-bases. We develop the query language in two steps. In this paper, we discuss the syntax of the language and devise a resolution based sequent style goal directed proof procedure, and thus take a proof theoretic approach to assigning semantics to our programs. As a next step, we develop (in a companion paper [17]) the notion of satisfaction for our language in Herbrand interpretation structures and define a model theory. We also develop a constructive way of computing models of programs by defining a fixpoint theory. Furthermore, we show that these three characterizations of this language are equivalent, and thus prove that the proof theory we develop here is indeed sound and complete. Completeness of the proof theory is a significant result by itself and has immense theoretical importance. Preliminary results are already at hand and may be found in a draft technical report [17]. It can be shown that the classical Horn clause logic and logics with overriding are special cases of our language.

The form of parameterization we consider is similar to the approach followed by van Emden [37], and Lakshmanan and Sadri [23]. We consider rules of the form $o : \mathcal{A} \leftarrow \mathcal{G} < \mu, \tau >$ to define a method in object o with mode μ, and type τ, in addition to rules of the form $\mathcal{A} \leftarrow \mathcal{G}$. Depending on the mode μ, the clause $\mathcal{A} \leftarrow \mathcal{G}$ will override a clause defining the same predicate in a superclass of o, or it will be unioned with the superclass clause(s) at o. Hence, the mode captures the relationship of the clauses defined in an object with the clauses defined higher in the hierarchy. But depending on the type, either the value $\mathcal{A}$, or the code (clause) $\mathcal{A} \leftarrow \mathcal{G}$ will be inherited in a subclass or instance of o, if at all. If the type is null, nothing will be inherited. Observe that the type τ relates o to its subclasses and instances with respect to the form of inheritance of the clause. Note that the inheritability of the clause $o : \mathcal{A} \leftarrow \mathcal{G} < \mu, \tau >$ in a subclass or instance q of o depends on the modes with which q defines predicates corresponding to the same method.

1.3 Organization of the Paper

In section 2, we present our parametric language by giving the syntax and discussing the informal semantics using an example in section 3. In this example, we introduce the salient features of the language and their functionalities. In section 4, we present a goal directed sequent style proof procedure for our language. Some results, in the form of properties of the language, are discussed in section 5. Section 6 discusses research along similar lines and compares our language with representative frameworks and languages. We then conclude in section 7.

2 The Parametric Language

In this section, we present the syntax of the language $\mathcal{L}$. The syntax is an extension of the standard Horn clause logic to maintain compatibility and hence, we take a relational approach to modeling the concept of methods in object-oriented paradigm instead of a functional approach as in F-logic.

The alphabet of our language $\mathcal{L}$ is a 6-tuple $\langle \mathcal{P}, \mathcal{O}, \mathcal{E}, \mathcal{V}, \mathcal{M}, \mathcal{T} \rangle$, where – (i) $\mathcal{P}$ is an infinite set of predicate symbols, (ii) $\mathcal{O}$ is an infinite set of object

symbols, including the distinguished object symbols $o^\top$ and *global*, (iii) $\mathcal{E}$ is an infinite set of function symbols where 0-ary function symbols serve the role of constants in $\mathcal{L}$, (iv) $\mathcal{V}$ is an infinite set of variables, (v) $\mathcal{M}$ is a finite set of symbols one each for every inheritance mode, and finally (vi) $\mathcal{T}$ is a finite set of symbols for inheritance types, one each for every inheritance type allowed in the language. These symbols are assumed to be pairwise disjoint. In this paper, we take the sets $\mathcal{M}$ and $\mathcal{T}$ to be {*override, inflate*} and {*value, code, null*} respectively. The set $\mathcal{I}$ of terms of $\mathcal{L}$ are defined in the usual manner from $\mathcal{O}$, $\mathcal{E}$ and $\mathcal{V}$. Intuitively, the set of terms $\mathcal{I}$ represents all possible object identifiers and values in $\mathcal{L}$. The alphabet also includes the usual logical connectives, quantifiers, and punctuation symbols. Throughout the paper, we use strings starting with uppercase letters for variables, lowercase letters for constants. Bold lowercase letters are used to denote arbitrary terms.

Atomic and Complex Formulas: The language of $\mathcal{L}$ consists of five basic types of atoms: (i) if $p \in \mathcal{P}$[1] and $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k$ are terms, then $p(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k)$ is a predicate or *p-atom*, (ii) if $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$ are object identifiers in $\mathcal{I}$, then $\boldsymbol{o}_1 :: \boldsymbol{o}_2$, $\boldsymbol{o}_1 \in \boldsymbol{o}_2$ and $\boldsymbol{o}_1 \preceq \boldsymbol{o}_2$, respectively, are subclass is-a, instance is-a and transitive is-a atoms; if $\boldsymbol{o}$ is an object id in $\mathcal{I}$, and $p(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k)$ is a p-atom then (iii) $global \ll p(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k)$ is a global atom, or *g-atom*, (iv) $\boldsymbol{o} \ll p(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k)$ is a message goal, or *m-atom*, and finally (v) $\boldsymbol{o} : p(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k)$ is a *local* atom, or *l-atom*. We use the notation $pred(\mathcal{A}) = p/k$ to denote the predicate symbol p of arity k when $\mathcal{A}$ is of the form $p(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k)$, $\boldsymbol{o} \ll p(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k)$, $global \ll p(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k)$, or $\boldsymbol{o} : p(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k)$. Also, we use the notation $\sharp$ for either ::, or $\in$, and $\natural$ for either ::, $\in$ or $\preceq$ in places where the distinctions between them are unimportant.

The formulas of $\mathcal{L}$ are defined as usual. Following the standard practice in logic programming, we consider only the definite Horn clause fragment of our language. A clause in $\mathcal{L}$ is of the form $\mathcal{A} \leftarrow \mathcal{B}_1, \ldots, \mathcal{B}_m < \mu, \tau >$, where $\mathcal{A}$ is called the head atom and $\mathcal{B}_i$s are called the body literals. While $\mathcal{A}$ can only be a p-, is-a or a l-atom, $\mathcal{B}_i$s can be any atom in $\mathcal{L}$ except an l-atom. μ and τ are empty for a clause if the head atom is a non l-atom, or is empty. A query is a clause without a head. A clause of the form $\mathcal{A} \leftarrow \mathcal{G}$ is called an *is-a clause* or *p-clause* if $\mathcal{A}$ is an is-a or p-atom, respectively; otherwise it is an *l-clause*. We use l-clauses to define properties – attributes and methods – of class or instance objects. In an l-clause, $\boldsymbol{o}$ is called the *context* of the method $pred(\mathcal{A})$ defined by the l-clause where $\mathcal{A} = \boldsymbol{o} : p(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k)$. For any given l-clause cl, or any l-atom $\mathcal{A}$, $context(cl)$ and $context(\mathcal{A})$ captures respectively the context of the l-clause or the l-atom.

Definition 2.1 (Programs and Queries) A program $\mathbf{P}$ in $\mathcal{L}$ is a triple $\langle \Lambda, \Pi, \mathcal{Q} \rangle$, where Λ is a set (possibly empty) of is-a and p-clauses, Π is a set (possibly empty) of l-clauses, and finally $\mathcal{Q}$ is a set (possibly empty) of queries. A program $\mathbf{P}$ is p-stratified if no is-a clause in Λ depends on the properties defined in the objects in the program $\mathbf{P}$, i.e., the body literals of is-a clauses do not contain m-atoms, and no clause in $\mathbf{P}$ contains the distinguished object $o^\top$. □

In the remainder of the paper, we consider only p-stratified programs.

3 Informal Semantics of Parametric Inheritance

At this point, it is perhaps beneficial to discuss some aspects of the language and the proof theory on more intuitive grounds to better understand the intricacies and the interplay among its various features. We exemplify our ideas of parametric inheritance using a simple university example. But before we proceed, certain points need further clarification.

We identify two types of l-clauses in our programs – *local* and *inherited* clauses. Intuitively, an l-clause is local to an object o, if it is defined in o, whereas it is inherited in o from q, if it is defined in q, and no intermediate superclass r of o (o inclusive) defines another clause for the same property in an overriding mode. We make these notions precise in the following two definitions.

Definition 3.1 (Locality) Let $\mathbf{P}$ be a program, $|\mathbf{P}|$ be its ground closure, $cl \equiv o : \mathcal{A} \leftarrow \mathcal{G} < \mu, \tau > \in |\Pi|$, and $context(cl) = o$. Then cl is local to every object o such that cl holds. We say that cl belongs to the program $\mathbf{P}(o)$. Hence, $\mathbf{P} = \bigcup_i \mathbf{P}(o_i) \cup \mathbf{P}(o^\top)$ where o_is are all the objects in $\mathbf{P}$, $o^\top$ is a distinguished object in $\mathcal{L}$ but not part of the program universe of $\mathbf{P}$, and $\mathbf{P}(o^\top) = \Lambda \cup \mathcal{Q}$. □

Definition 3.2 (Inherited Clauses) Let $\mathbf{P}$ be a program, $|\mathbf{P}|$ be its ground closure, $cl \equiv \mathcal{A} \leftarrow \mathcal{G} < \mu, \tau > \in |\Pi|$, and $context(cl) = o$. Then cl is inherited in q if ($q \preceq o$ hold, $\tau \neq null$, $\neg\exists r$ such that ($r \neq o$, $q \sharp r$, $r \preceq o$[2] hold, and $\exists cl' \in |\Pi|$, such that $cl' \equiv \mathcal{A}' \leftarrow \mathcal{G}' < override, \tau' >$, $pred(\mathcal{A}) = pred(\mathcal{A}')$ and $context(cl') = r$)). If cl is inherited in q and

[1] By the notation p/k we denote a predicate p of arity k in this paper.

[2] Including $r = q$, essentially implying that q does not have a local definition for $pred(\mathcal{A})$ that overrides cl.

τ = *code* then we say *cl* is *code inherited* in q, else if τ = *value*, *cl* is *value inherited* in q. □

If q code inherits *cl* then $\mathcal{A}$ holds in q only if $\mathcal{G}$ holds in q. On the other hand, if q value inherits *cl* then $\mathcal{A}$ holds in q only if $\mathcal{G}$ holds in o. Note that, clause *cl* is not inherited if τ = *null*. Also the effect of inflation is somewhat built-in in the above definition because if any intermediate object r in between o and q defines a clause for $pred(\mathcal{A})$ in a mode other than overriding, it does not affect the inheritability, effectively capturing the notion of inflation. With the definitions as essential prerequisites, we now explain the functionalities of our language with an example.

Example 3.1 Consider the program $\mathbf{P}_1$ in figure 3, and the corresponding schematic representation in Figure 2 for easy reference[3]. We must mention that only the immediate is-a (subclass or instance) clauses are specified by the users. The transitive is-a relationships may be easily computed using a set of axioms added to each program. Users also implicitly assume the locality and inheritability of clauses according to definitions 3.1 and 3.2 and expect that the system will take care of inheritance according to the specifications contained in the program.

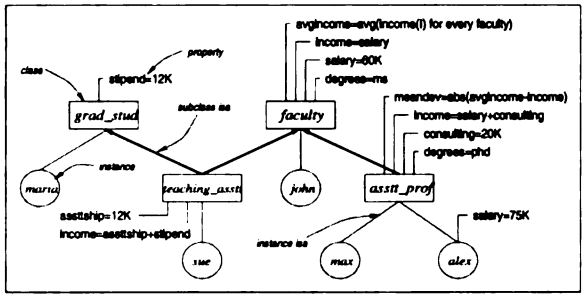


Figure 2: Schematic representation of the university database.

In this example, the class object faculty locally defines methods (or properties) *degrees, salary, income,* and *avgincome* through clauses 8 through 11. These properties are available for inheritance at the classes asstt_prof and teaching_asstt as they are defined as subclasses of faculty (clauses 1 and 2). While asstt_prof defines two new properties *meandev* and *consulting* (clauses 13 and 15), it also redefines *income* and *degrees* (clauses 12 and 14). Since *degrees* is redefined in inflating mode, astt_prof will inherit the value (consequent) of clause 8 from faculty[4]. Notice that we expect max's *degrees* to be the set {ms, phd}, not {phd} alone (query (20)). This is because max's closest ancestor asstt_prof class redefines *degrees* in inflating mode and hence, the values specified in asstt_prof must be added to the inherited values from the class object faculty.

But the definition for *income* overrides clause 10 in faculty as clause 14 is defined in an overriding mode. Hence, clause 10 will not be applicable in asstt_prof. On the other hand, alex being an instance of asstt_prof (clause 6), and not having defined any local definition for *income*, inherits clause 14. This time it inherits the code, i.e., $income(X) \leftarrow salary(Y), consulting(Z), X = Y + Z$. The effect of this is that $income(95K)$ holds (query (22)) at alex since the salary for alex is 75K which overrides the class property salary in clause 9. This is similar to the idea of dynamic self (late) binding in object-oriented systems. Notice that max will have only 80K as his *salary* (query (23)). Following a similar analysis, it is easy to see that john's salary is only 60K. Similar remarks apply to sue and maria with respect to their *incomes*.

Now consider the *meandev* clause (clause 15) defined in asstt_prof class. Since this clause is code inherited in every subclass/instance of asstt_prof which in turn uses the *avgincome* defined in faculty (clause 11), we expect that *meandev* (query (24)) for alex will be 34.57, and for max, 19.57, since the *avgincome* (query (25)) computed in faculty is 60.43 (inclusive of the class object values at faculty, teaching_asstt, and asstt_prof). Note that we value inherit *avgincome* in asstt_prof and then in all its instances from faculty. Had we code inherited *avgincome* instead, we would have had different *avgincomes* at different levels in the object hierarchy, and the *meandev* computed at any level would have been different. □

4 Proof Theory

In this section, we develop a proof procedure in order to assign operational semantics to programs in our language. We present a goal directed sequent style proof system similar to Miller's module language [27], Contextual Logic Programming [28], Selflog [5], OR-Log [19], etc. The proof system is defined as a set of properties for the proof predicate $\vdash$. In this proof system, goals are proven in the context of a pair of objects, called the initial context and the current context (generally called the context or the context pair). The purpose of contextualizing provability will become clear as we discuss the proof system and, in particular, when we discuss an example proof later in this

[3]Notice that the scheme in figure 2 and the program in figure 3 is an adaptation of the scheme in figure 1.

[4]Because clause 8 is defined as a value inheritable clause. In this case, it does not really matter as it is a fact, but for methods defined via rules, it does make a difference, as we will see later.

$\Lambda_1 :=$

(1) $asstt\_prof :: faculty.$
(2) $teaching\_asstt :: faculty.$
(3) $teaching\_asstt :: grad\_stud.$
(4) $john \in faculty.$
(5) $max \in asstt\_prof.$
(6) $alex \in asstt\_prof.$
(7) $sue \in teaching\_asstt.$
(8) $maria \in grad\_stud.$

$\Pi_1 :=$

(8) $faculty : degrees(ms) \leftarrow < override, value > .$
(9) $faculty : salary(60K) \leftarrow < override, value > .$
(10) $faculty : income(X) \leftarrow salary(X)$
$< override, code > .$
(11) $faculty : avgincome(X) \leftarrow O \ll income(Y),$
$O \preceq faculty, X = avg(< Y >)$
$< override, value > .$
(12) $asstt\_prof : degrees(phd) \leftarrow < inflate, value > .$
(13) $asstt\_prof : consulting(20K) \leftarrow$
$< override, value > .$
(14) $asstt\_prof : income(X) \leftarrow salary(Y),$
$consulting(Z), X = Y + Z$
$< override, code > .$
(15) $asstt\_prof : meandev(X) \leftarrow income(Y),$
$avgincome(Z), X = abs(Z - Y)$
$< override, code > .$
(16) $grad\_stud : stipend(12K) \leftarrow < override, value > .$
(17) $teaching\_asstt : income(X) \leftarrow assttship(Y),$
$stipend(Z), X = Y + Z$
$< override, code > .$
(18) $teaching\_asstt : assttship(12K) \leftarrow$
$< override, value > .$
(19) $alex : salary(75K) \leftarrow < override, value > .$

$\mathcal{Q}_1 :=$

(20) $? \ max \ll degrees(X).$
(21) $? \ max \ll income(X).$
(22) $? \ alex \ll income(X).$
(23) $? \ O \ll income(X).$
(24) $? \ O \ll meandev(X),$
$O \in asstt\_prof.$
(25) $? \ faculty \ll avgincome(X).$

Figure 3: University database in $\mathcal{L}$.

(EMPTY) $\dfrac{}{\mathbf{P}, \langle \boldsymbol{p}, \bar{o} \rangle \vdash_\epsilon \Box}$

(MESSAGE) $\dfrac{\mathbf{P}, \langle o, o \rangle \vdash_\theta p(a_1, \ldots, a_k)}{\mathbf{P}, \langle \boldsymbol{p}, \bar{o} \rangle \vdash_\theta o \ll p(a_1, \ldots, a_k)}$

(AND) $\dfrac{\mathbf{P}, \langle \boldsymbol{p}, \bar{o} \rangle \vdash_\theta \mathcal{G}_1 \qquad \mathbf{P}, \langle \boldsymbol{p}, \bar{o} \rangle \vdash_\sigma \mathcal{G}_2[\theta]}{\mathbf{P}, \langle \boldsymbol{p}, \bar{o} \rangle \vdash_{\theta\sigma} \mathcal{G}_1, \mathcal{G}_2}$

(TOP) $\dfrac{\mathbf{P}, \langle o^\top, o^\top \rangle \vdash_\theta \mathcal{G}}{\mathbf{P} \vdash_\theta \mathcal{G}}$

(GLOBAL) $\dfrac{\mathbf{P}, \langle o^\top, o^\top \rangle \vdash_\sigma \mathcal{A}}{\mathbf{P}, \langle \boldsymbol{p}, \bar{o} \rangle \vdash_\sigma \mathcal{A}} \quad \left(\begin{array}{l} \mathcal{A} \text{ is a g- or is-a atom,} \\ \neg\exists\psi, \psi = mgu(o^\top, \boldsymbol{p}) \end{array} \right)$

(DEDUCTION) $\dfrac{\mathbf{P}, \langle o^\top, o^\top \rangle \vdash_\sigma \mathcal{G}[\theta]}{\mathbf{P}, \langle o^\top, o^\top \rangle \vdash_{\theta\sigma} \mathcal{A}} \quad \left(\begin{array}{l} \mathcal{A}' \leftarrow \mathcal{G} \in \mathbf{P}, \\ \theta = mgu(\mathcal{A}, \mathcal{A}') \end{array} \right)$

(SELF) $\dfrac{\mathbf{P}, \langle \boldsymbol{p}, \boldsymbol{p} \rangle \vdash_\sigma \mathcal{G}[\theta\phi]}{\mathbf{P}, \langle \boldsymbol{p}, \boldsymbol{p} \rangle \vdash_{\theta\phi\sigma} p(a_1, \ldots, a_k)} \quad \left(\begin{array}{l} \neg\exists\psi, \psi = mgu(\boldsymbol{p}, o^\top), \theta = mgu(o, \boldsymbol{p}), \\ o[p(a'_1, \ldots, a'_k) \leftarrow \mathcal{G} < \mu, \tau >] \in \mathbf{P}, \\ \phi = mgu(< a_1, \ldots, a_k >, < a'_1, \ldots, a'_k > [\theta]) \end{array} \right)$

(CODE SHARING) $\dfrac{\mathbf{P}, \langle \boldsymbol{p}, \boldsymbol{p} \rangle \vdash_\sigma \mathcal{G}[\theta\phi]}{\mathbf{P}, \langle \boldsymbol{p}, \bar{o} \rangle \vdash_{\theta\phi\sigma} p(a_1, \ldots, a_k)} \quad \left(\begin{array}{l} \neg\exists\psi, \psi = mgu(\bar{o}, \boldsymbol{p}), \theta = mgu(o, \bar{o}), \\ o : p(a'_1, \ldots, a'_k) \leftarrow \mathcal{G} < \mu, code > \in \mathbf{P}, \\ \phi = mgu(< a_1, \ldots, a_k >, < a'_1, \ldots, a'_k > [\theta]) \end{array} \right)$

(VALUE SHARING) $\dfrac{\mathbf{P}, \langle o, o \rangle \vdash_\sigma \mathcal{G}[\theta\phi]}{\mathbf{P}, \langle \boldsymbol{p}, \bar{o} \rangle \vdash_{\theta\phi\sigma} p(a_1, \ldots, a_k)} \quad \left(\begin{array}{l} \neg\exists\psi, \psi = mgu(\bar{o}, \boldsymbol{p}), \theta = mgu(o, \bar{o}), \\ o : p(a'_1, \ldots, a'_k) \leftarrow \mathcal{G} < \mu, value > \in \mathbf{P}, \\ \phi = mgu(< a_1, \ldots, a_k >, < a'_1, \ldots, a'_k > [\theta]) \end{array} \right)$

(INHERITANCE) $\dfrac{\mathbf{P}, \langle \boldsymbol{p}, \bar{o} \rangle \vdash_\sigma \bar{o} \sharp R \qquad \mathbf{P}, \langle \boldsymbol{p}, R \rangle \vdash_\theta \mathcal{A}[\sigma]}{\mathbf{P}, \langle \boldsymbol{p}, \bar{o} \rangle \vdash_{\sigma\theta} \mathcal{A}} \quad \left(\begin{array}{l} \neg\exists\psi, \psi = mgu(o, \bar{o}[\sigma]), pred(\mathcal{A}) = pred(\mathcal{A}'), \\ o : \mathcal{A}' \leftarrow \mathcal{G} < override, \tau > \in \mathbf{P} \end{array} \right)$

Figure 4: Proof rules.

section. Intuitively, contexts are used to memorize the objects where we started proving a goal (mainly property goals), and in which object we are currently attempting a proof in the style of a *memory stack*. The combination of these two contexts gives us the capability to develop a robust proof procedure with all its intricate inheritance modes and types and the complex interplay among them. In [6], a similar but simpler context-based proof procedure was used to define provability for logic programs with encapsulation.

Similar to [28] and [6], we read the proof rules from bottom up. We use the notation $\mathbf{P}, \langle p, \bar{o}\rangle \vdash_\theta \mathcal{G}$ to represent the fact that the goal $\mathcal{G}$ is derivable from the program $\mathbf{P}$ with an initial context p, a current context $\bar{o}$, and with a substitution θ, i.e., $\mathbf{P}, \langle p, \bar{o}\rangle \vdash \mathcal{G}\theta$. Note that the application of each proof rule is contingent upon satisfaction of the conditions specified on the right hand side of each rule (if any). We use θ, ϕ, σ, ψ, etc. to represent most general unifiers, and ϵ to represent the empty substitution. $\Box$ denotes the empty goal, which is always true in any context.

As usual, a proof for $\mathbf{P} \vdash \mathcal{G}\theta$ is a tree, called the *proof tree*, rooted at $\mathbf{P} \vdash_\theta \mathcal{G}$ with internal nodes that are instances of one of the following rules except TOP, and with leaf nodes that are labeled with the figure EMPTY. The *height* of a proof is the maximum of the number of nodes in all the branches in the proof tree, and the *size* of a proof is the number of nodes in the proof tree. But in contrast to [28],[5] and [6], the substitutions in our proof rules must also be uniformly applied to the initial and current contexts to correctly capture the intended operational semantics. That is, a proof $\mathbf{P}, \langle p, \bar{o}\rangle \vdash_\theta \mathcal{G}$ or $\mathbf{P}, \langle p, \bar{o}\rangle \vdash \mathcal{G}\theta$ actually means $\mathbf{P}, \langle p, \bar{o}\rangle\theta \vdash \mathcal{G}\theta$ in our system.

The set of proof rules for our language is shown in figure 4. The first four rules – EMPTY, AND, MESSAGE and TOP – are among the most basic ones. The interpretation of the provability relation $\vdash$ is simple, though not very immediate. We explain these rules below to clarify the idea.

Consider the rule TOP first. This rule says, to prove any goal $\mathcal{G}$ from the program $\mathbf{P}$, we must prove it from $\mathbf{P}$ by letting both the initial and current contexts, called the *context pair*, to be $o^\top$. Recall that, $o^\top$ is a special object in $\mathcal{L}$, but it is not part of any program universe. This rule initiates the proof in a contextual manner. So, for every proof, TOP is the first rule to be fired, i.e., the root is labeled TOP, and this rule is never applied afterwards in a proof. Intuitively, the rule provides the default object $o^\top$ as the context pair for proving $\mathcal{G}$ in a global context (outside of any object). If the goal is conjunctive, it can be proven individually in any unspecified order by splitting the goal using the AND rule with respect to the identical context pair. The EMPTY rule says that the empty goal $\Box$ is provable with empty substitution from any context pair at any time.

Any is-a or g-goal can be proven by using the DEDUCTION rule. Notice that we can never have an l-atom, as it never appears in the body of any clause, including queries. Also, we will never fire this rule for any message goal as the goal will never unify with the head of any clause since m-atoms do not appear in the consequents, making this rule non ambiguous. Instead, we fire the MESSAGE rule for m-goals. Notice that other than message goals, all goals must be proven outside any object in a global context, and hence application of DEDUCTION is justified.

Consider proving a predicate (representing a property of the object) in an object. We must send a message to the object to evaluate the predicate using m-goals of the form $o \ll p(a_1, \ldots, a_k)$. To prove such a goal, we apply the message rule and initialize the context pair as $\langle o, o\rangle$ indicating that we must prove $p(a_1, \ldots, a_k)$ in the context of o, and that the initial context we started with is also o. The initial context is meant to capture the entry point in the object hierarchy.

We now have basically four distinct ways to prove $p(a_1, \ldots, a_k)$ in the current context, namely $\bar{o}$, if the context pair is $\langle p, \bar{o}\rangle$. (i) If we have an l-clause at the initial context p, then we can prove the goal using the SELF rule. This would allow proving the goal using a local l-clause independent of its mode and type. (ii) If a local definition at the initial context p is not found, then we go one level up the hierarchy firing rule INHERITANCE such that $\bar{o} \sharp R$ holds[6], and we now try to prove the goal in the context of $\langle p, R\rangle$. Note that if we now find an l-clause at R, we are able to copy the code at p if the clause is code inheritable using the rule CODE SHARING in the next step. We can do so because the initial context , p, was memorized. We can also value inherit the l-clause in the same way, but use R as the context instead of $\bar{o}$. It is important to note that we can fire INHERITANCE only if we do not have an l-clause at $\bar{o}$ in overriding mode. This essentially captures overriding and inflation – we inherit a clause if there is no overriding or if we are allowed to inflate. We continue to climb up the hierarchy until we fail, or we find an l-clause that we can use to prove the goal.

Once we climb up at any level $\bar{o}$ in the hierarchy, and

[5]The authors also used substitutions in contexts in a later work on contextual logic programming.

[6]Notice that initially the contexts p and $\bar{o}$ are identical.

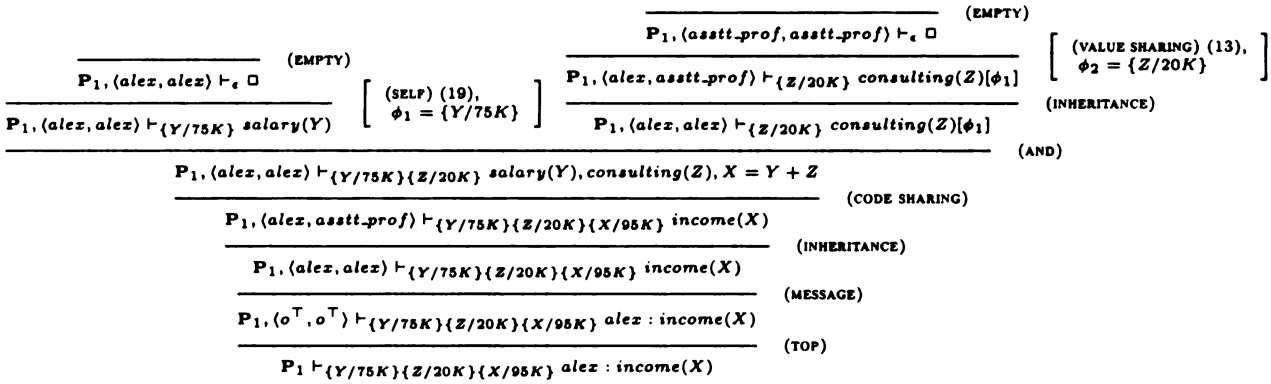


Figure 5: A proof tree for the goal $\mathbf{P}_1 \vdash alex : income(X)$

if the object $\bar{o}$ has a local definition, either (iii) the VALUE SHARING, or (iv) the CODE SHARING rule will be fired depending on the type of the l-clause. If the rule VALUE SHARING is fired, according to the definition of value sharing, we must then evaluate the body of the clause in the context of $\bar{o}$, the current context. Hence, we must initialize the context pair to $\langle \bar{o}, \bar{o}\rangle$. We will initialize it to $\langle p, p\rangle$ if we fired code sharing since the clause's body must now be proven in the initial context, essentially copying the body of the clause at the initial context.

But once inside a context, we must be able to prove is-a and global predicate goals. So, the rule GLOBAL helps us to return to the global context. We do not reenter a context until we need to prove another message goal. Notice that goals of the form $global \ll p(a_1, \ldots, a_k)$ are a technicality. Without such specialized syntax, we have no way simple or intuitive way of distinguishing a global predicate from an object property, and hence will never come out of the context of an object thereby resulting in a failed proof, or prove a predicate defined in the object which has the same name p/k.

We expect that it is clear from the above discussion how modes and types affect the provability of a goal. Essentially, the modes affect the inheritance (firing of rule INHERITANCE depends on the mode) of clauses at any object, while the type affects the evaluation of the clause through the initialization of the context pair (dictates where the code must be evaluated). Notice that the null inheritance is a by-product of the rules SELF, CODE INHERITANCE and VALUE INHERITANCE, and the way they are applied.

Example 4.1 (An Example Proof) Consider the example 3.1 in section 3. The proof tree in figure 5 shows a sample proof for $\mathbf{P}_1 \vdash alex : income(X)$ using the proof rules in section 4. Note that we do not show, for want of space, the branch corresponding to $\mathbf{P}_1, \langle alex, alex\rangle \vdash_\sigma alex \sharp R$ in every application of the INHERITANCE rule, which succeeds with a binding $\{R/asstt\_prof\}$ through a subsequent application of IS-A DEDUCTION with clause (6). Recall that in section 3, we anticipated exactly the same value of income for *alex* as returned here by the proof procedure, i.e., 95K.

5 Properties of the Language

The operational semantics based on the proof theory which we have presented in the previous section has several interesting properties. In this section, we highlight some of those properties for expository reasons, but simultaneously avoiding extensive elaborations. The exact details may be found in a companion paper [17]. First, consider adding an additional clause (20), $grad\_stud : salary(25K) \leftarrow\!< override, value >$ in program $\mathbf{P}_1$ in example 3.1. In this case, using our proof rules, it is possible to prove $\mathbf{P}_1 \vdash sue \ll salary(25K)$, and also $\mathbf{P}_1 \vdash sue \ll salary(60K)$. This is so because, teaching_asstt is a subclass of both faculty and grad_stud, and *salary* is inheritable from both grad_stud and faculty according to definition 3.2 and also by the application of the inheritance rules in the proof system. This is not exactly what is desired. What is more meaningful is that the proof system finds exactly one of these two proofs, not both. That is, $\mathbf{P}_1 \vdash sue \ll salary(60K), sue \ll salary(25K)$ must fail. This is a consequence of allowing multiple inheritance. Most logic based and many procedural languages, disallow multiple inheritance to avoid just such

situations, and the even more serious issue of potential inheritance conflict. Our solution to this problem is to disallow programs that incorporate this type of ambiguity. Thus, we allow a much larger set of programs, yet maintain uniqueness of proofs. The following definitions and theorems help in achieving this goal.

Definition 5.1 Let $\mathbf{P}$ be a p-stratified program, and $|\mathbf{P}|$ be its ground closure. $\mathbf{P}$ is *conflict free* if for all objects o in $|\mathbf{P}|$, for every property p/k inheritable in o from some class r, r is unique. □

Theorem 5.1 Let $\mathbf{P}$ be a conflict free program, and $\mathcal{A}$ be any message goal. Then the proof $\mathbf{P} \vdash \mathcal{A}$ is unique.

Proof Sketch: Follows immediately from the definition of inheritability and from the fact that the object hierarchy essentially behaves like a forest with respect to inheritability of a given method in an object. □

We can show that the Horn fragment of predicate logic is a special case of our language, and that any proof for any goal in predicate logic will only use the rules EMPTY, DEDUCTION, AND, and TOP. We can easily prove the following theorem.

Theorem 5.2 Let P be a Horn program in predicate logic, and let $\vdash^*$ be the corresponding provability relation. Let $\mathbf{P}'$ be the conflict free program in $\mathcal{L}$ such that $\mathbf{P}'(o^\top) = P$ and $\Lambda' = \emptyset$, and $\Pi' = \emptyset$. Then, for any goal $\mathcal{G}$, $P \vdash^* \mathcal{G}[\theta] \iff \mathbf{P}' \vdash \mathcal{G}[\theta]$.

Proof Sketch: Showing that the proof trees are identical except for the fact that TOP is the only additional rule applied in $\vdash$, and has a height $k+1$ if the height in $\vdash^*$ is k. □

It is also possible to define satisfaction, interpretation structures, and a model theory. We can show that a least model exists for every conflict free program. Furthermore, a fixpoint operator can be defined to construct the least model as the least fixpoint of that operator by showing that the fixpoint operator is monotonic. Finally, the completeness of the proof system can be proven using the equivalence of least models, fixpoint, and provability through the following theorems.

Theorem 5.3 Let $\mathbf{P}$ be a program and $\mathbf{M_P}$ be its least model. Then, $\mathbf{M_P} = lfp(\mathbf{T_P}) = \mathbf{T_P} \uparrow^\omega$.

Theorem 5.4 (Equivalence) Let $\mathbf{P}$ be a program, $\mathbf{M_P}$ be its least model, and $\mathcal{G}$ be a ground goal. Then, we have $\mathbf{P} \vdash_\epsilon \mathcal{G} \iff \mathbf{M_P} \models \mathcal{G}$ □

6 Comparison with Related Research

A critical observation we have made is that immediate is-a and transitive is-a relations need to be treated differently. This observation has enabled us to define the notions of *local* and *inherited* clauses, which we have successfully utilized to capture inheritance within a sound and complete proof theory in a natural and elegant way. Almost no other language makes this distinction and consequently, suffers from several drawbacks. As a result, some of those proposals [21, 12] take a simpler approach to inheritance modeling by adopting value inheritance and avoiding the complicated issue of code re-use. Selflog [5] attempted to address the issue of code re-use in a limited way but did not capture value inheritance. Similar remarks apply to ORLog [19] which also captured only code re-use, but not value inheritance. Although F-logic takes a value inheritance approach, code re-use can be simulated using the higher-order features of the language in a limited way, which appears to be unnatural and nonintuitive.

Consider the following program $\mathbf{P}_2 = \langle \Lambda_2, \Pi_2, \emptyset \rangle$ adapted from [7, 21], where $\Sigma_2 = \emptyset$. By declaring it meaningless, the stable semantics in [7] assigns no meaning to this program. F-logic assigns the only minimal model $I =< I(o), I(c), I(d), I(o^\top) >$ such that $I(o) = \{p(a)\}$, $I(c) = \{p(b)\}$, $I(d) = \{p(a)\}$ and $I(o^\top) = \{o \in c, o \in d, c :: d\}$. In contrast, our semantics, if the program is allowed[7], will assign the following least model to $\mathbf{P}_2$: $I' =< I'(o), I'(c), I'(d), I'(o^\top) >$ such that $I'(o) = \{p(a), p(b)\}$, $I'(c) = \{p(b)\}$, $I'(d) = \{p(a)\}$ and $I'(o^\top) = \{o \in c, o \in d, c :: d\}$. The reason for this striking difference is that [7] views c as an intermediate object in between o and d[8], and also since it makes no distinction between immediate and transitive is-a. In contrast, we view d and c as two distinct (immediate) superclasses of o, and hence, the difference. In F-logic, on the other hand, $p(a)$ is (value) inherited in o first from d and hence, overrides any future inheritance (from c) with respect to $p/1$.

$$\Lambda_2 := \left| \begin{array}{ll} (1) & o \in c \leftarrow o \ll p(a). \\ (2) & c :: d \leftarrow o \ll p(a). \\ (3) & o \in d. \end{array} \right.$$

$$\Pi_2 := \left| \begin{array}{ll} (4) & d : p(a) \leftarrow < override, code > . \\ (5) & c : p(b) \leftarrow < override, code > . \end{array} \right.$$

Observe that virtual sub-classing (subclass definitions

[7] Notice that this is not a p-stratified program.

[8] This is a consequence of the definition of local and inherited clauses adapted in [7].

using is-a rules) is captured naturally in our language and is not an issue because of the distinctions made between immediate and transitive is-a. In ORLog, o does not inherit $p/1$ from either c or d since it treats the situation as an inheritance conflict. Our language cannot recognize inheritance conflicts and behaves similarly to [12] and [7]. F-logic, on the other hand, settles for multiple minimal models in the event of inheritance conflicts.

Again consider the following program $\mathbf{P}_3$ in $\mathcal{L}$.

$$\Lambda_3 := \left| \begin{array}{ll} (1) & o \in c \leftarrow . \\ (2) & c :: d \leftarrow . \end{array} \right.$$

$$\Pi_3 := \left| \begin{array}{ll} (3) & d : p(a) \leftarrow < override, code > . \\ (4) & c : p(X) \leftarrow r(X) < override, code > . \\ (5) & c : p(c) \leftarrow < inflate, value > . \\ (6) & c : r(b) \leftarrow < override, code > . \\ (7) & o : r(d) \leftarrow < override, code > . \end{array} \right.$$

Observe that although clause (5) defines $p/1$ in inflating mode, neither c nor o inherits $p(a)$ from d since one clause in c corresponding to $p/1$ (clause (4)) is enough to override it. Furthermore, $p(c)$ is derived in o by value inheriting clause (5) in c, but o derives $p(d)$ by code inheriting clause (4) from c, and o does not derive $p(b)$ in particular. Note that in this instance o is inheriting the definition of $p/1$ from c in two ways, using inheritance type code and value. This is not possible in any other language. However, as can be seen from the example 3.1, inheritance withdrawal is not possible in our language, while it is (only) possible in ORLog [19]. For instance, *salary* and *consulting* values are being inherited in sue from asstt_prof and faculty classes of which it is an instance. This may not be desired in some applications. In particular, $\mathbf{P}_1 \vdash sue \ll salary(X)$ is provable with a binding $X = 60K$, which is probably not intended, but becomes possible anyway due to (perhaps unwanted) inheritance. Observe that defining *salary* as *null* type at faculty does not really help, since now asstt_prof class and its instances cannot inherit *salary* from faculty either. In that sense, ORLog appears to handle conflicts better than our language, at least in this regard.

We must confess that while our language captures inheritance in a parametric way which no other language does, we only consider Horn fragment of our language. F-logic, however, is much more general and provides a foundation for object-oriented languages. In fact, our current work is inspired by F-logic and is an attempt to capture the semantics of inheritance in a sound and complete proof theory that was missing in F-logic. We believe that our work has successfully exploited the restricted setting of our language to capture modes and types in a single logical framework as first class citizens, to our knowledge, for the first time. We have also been able to capture F-logic's notion of non-inheritable methods via null types. A further advantage of our language is that we can simulate almost all known inheritance modes and types in our language.

For the sake of brevity, we have not included a detailed comparison of our approach to inheritance modeling with those in the realm of artificial intelligence. This is also because most artificial intelligence approaches only deal with overriding or exception handling. However, an eloquent discussion on the differences between approaches in artificial intelligence and object-oriented communities may be found in [24].

7 Conclusion

In this paper, we have considered a simple and intuitive object-oriented logic and developed an elegant, resolution- based and sequent style goal directed proof procedure for its definite clause fragment in which we captured the semantics of parametric inheritance with multiple modes and types, and dynamic subclassing in a declarative way. The proof theory based on contextual provability is sound and complete with respect to its model theoretic semantics. We did not impose any restriction[9] on the class of programs. Nor did we require any pre-processing of the programs in capturing the provability of goals in our programs. The novelty of this proof system is that it captures the definition of inherited clauses and the meanings of modes and types declaratively.

The predicate oriented method implementation gives it a classical first-order logic flavor and makes it easier to understand the logic in a much understood syntax, and (partly) the semantics. In fact, this language can dual as a classical first-order language, as an object-oriented language, or both, as captured by the theorem 5.2. It is more likely that query optimization techniques developed for Datalog [36] can be tailored to apply in our framework. Although our language cannot directly account for functional methods as does F-logic and ORLog, we can utilize constraints to model them in a limited way.

Future work will consider extensions to our proposed language. Note that we have fixed ahead the set of allowed modes and types for our language. This may be relaxed by capturing modes and types as computable

[9] Except that we consider only conflict free (definition 5.1) Horn programs, which is not really a restriction on its expressibility. Rather, it is a semantic restriction to avoid ambiguity in the meanings of programs.

functions. In this approach, the users will supply the modes and types and define their meanings as part of their programs. The system will then define the semantics of the logic program by computing the meanings of modes and types accordingly. A modal semantics for our language can be developed to further understand its logical foundations, and to incorporate encapsulation. To demonstrate the feasibility of our language, we have implemented a variant of this language, called Datalog^{++}, on Unix and NT machines based on XSB [33, 32]. A web version of Datalog^{++} demo may be found at `http://www.cs.msstate.edu/~jamil`.

Acknowledgments

Research supported in part by a grant from the Department of Computer Science, Mississippi State University, and by the College of Engineering, Mississippi State University through a grant under the Research Initiation Program. The author would like to thank Durriya Meer of Wright State University, USA for giving helpful comments to improve the presentation of the initial version of the paper.

References

[1] S. Abiteboul, G. Lausen, H. Uphoff, and E. Waller. Methods and Rules. In *ACM SIGMOD Conference on Management of Data*, pages 32–41, 1993.

[2] H. Aït-Kaci and R. Nasr. LOGIN: a logic programming language with built-in inheritance. *Journal of Logic Programming*, 3:182–215, 1986.

[3] H. Aït-Kaci and A. Podelski. Towards a Meaning of LIFE. Technical Report 11, Digital Paris Research Labs, 1991.

[4] F. Belli, O. Jack, and L. Naish. Object-oriented programming in Prolog: Rationale and a case study. Technical Report 92/2, Department of Electrical and Electronics Engineering, University of Paderborn, Paderborn, Germany, 1992.

[5] M. Bugliesi. A declarative view of inheritance in logic programming. In K. Apt, editor, *Proc. Joint Int. Conference and Symposium on Logic Programming*, pages 113–130. The MIT Press, 1992.

[6] M. Bugliesi and H. M. Jamil. A logic for encapsulation in object oriented languages. In M. Hermenegildo and J. Penjam, editors, *Proceedings of the 6th International Symposium on Programming Language Implementation and Logic Programming (PLILP)*, pages 213–229, Madrid, Spain, 1994. Springer-Verlag. LNCS 844.

[7] M. Bugliesi and H. M. Jamil. A stable model semantics for behavioral inheritance in deductive object oriented languages. In G. Gottlob and M. Y. Vardi, editors, *Proceedings of the 5th International Conference on Database Theory (ICDT)*, pages 222–237, Prague, Czech Republic, 1995. Springer-Verlag. LNCS 893.

[8] J.S. Conery. Logical objects. In R. A. Kowalski and K. A. Bowen, editors, *Proc. 5th Int. Conference on Logic Programming*, pages 420–434. The MIT Press, 1988.

[9] Andrew Davidson. *A Survey of Logic Programming-based Object Oriented Languages.* Addison Wesley, 1992.

[10] P. Dechamboux and C. Roncancio. Peplom^{d}: An object-oriented database programming language extended with deductive capabilities. In D. Karagiannis, editor, *Proc. of the 5th Intl. DEXA Conf.*, pages 2–14, Athens, Greece, 1994. Springer-Verlag. LNCS 856.

[11] Y. Dimopoulos and Antonis Kakas. Logic programming without negation as failure. In John Lloyd, editor, *Proceedings of the 12th International Logic Programming Symposium*, pages 369–383, Portland, Oregon, December 1995. MIT Press.

[12] G. Dobbie and R. Topor. A Model for Sets and Multiple Inheritance in Deductive Object-Oriented Systems. In *Proc. 3rd Intl. DOOD Conf.*, pages 473–488, December 1993.

[13] K. Fukunaga and S. Hirose. An experience with a Prolog based object-oriented language. In *OOPSLA-86*, pages 224–231, Portland, Oregon, September 1986.

[14] H. Gallaire. Merging object and logic programming: Relational semantics. In *AAAI-86*, pages 754–758, Philadelphia, PA, 1986.

[15] Jiawei Han, Yongjian Fu, and Raymond T. Ng. Cooperative query answering using multiple layered databases. In *Proceedings of the 2nd International Conference on Cooperative Information Systems (CoopIS*, pages 47–58, Toronto, Canada, 1994.

[16] H. V. Jagadish, Laks V. S. Lakshmanan, and Divesh Srivastava. What can hierarchies do for data warehouses? In *Proc. of the VLDB Conference*, pages 530–541, 1999.

[17] H. M. Jamil. A logical framework for parametric inheritance. Technical report, Department of Computer Science, Mississippi State University, USA, January 2000. In preparation.

[18] H. M. Jamil and L. V. S. Lakshmanan. ORLog: A Logic for Semantic Object-Oriented Models. In *Proc. 1st Int. Conference on Knowledge and Information Management*, pages 584–592, 1992.

[19] H. M. Jamil and L. V. S. Lakshmanan. A declarative semantics for behavioral inheritance and conflict resolution. In John Lloyd, editor, *Proceedings of the 12th International Logic Programming Symposium*, pages 130–144, Portland, Oregon, December 1995. MIT Press.

[20] Hasan M. Jamil. Belief reasoning in mls deductive databases. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *Proceedings ACM SIGMOD International Conference on Management of Data, Philadephia, Pennsylvania*, pages 109–120. ACM Press, 1999.

[21] M. Kifer, G. Lausen, and J. Wu. Logical Foundations for Object-Oriented and Frame-Based Languages. *Journal of the Association of Computing Machinery*, 42(3):741–843, July 1995.

[22] E. Laesen and D. Vermeir. A Fixpoint Semantics for Ordered Logic. *Journal of Logic and Computation*, 1(2):159–185, 1990.

[23] L. V. S. Lakshmanan and F. Sadri. Probabilistic deductive databases. In *Proc. International Logic Programming Symposium*, pages 254–268, Ithaca, NY, November 1994. MIT Press.

[24] L. V. S. Lakshmanan and K. Thirunarayan. Declarative frameworks for inheritance. In Jan Chomicki and Gunter Saake, editors, *Logics for Databases and Information Systems*, pages 357–388. Kluwer Academic Publishers, 1998.

[25] Y. Lou and Z. M. Ozsoyoglu. An object-oriented deductive language with methods and method inheritance. In J. Clifford and R. King, editors, *Proceedings of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 198–207, Denver, Colorado, 1991.

[26] F.G. McCabe. *Logic and Objects*. Prentice Hall International, London, 1992.

[27] D. Miller. A Logical Analysis of Modules in Logic Programming. *Journal of Logic Programming*, 6(1/2):79–108, January/March 1989.

[28] L. Monteiro and A. Porto. Contextual Logic Programming. In *6th ALP Intl. Conf. on Logic Programming*, 1989.

[29] L. Monteiro and A. Porto. A transformational view of inheritance in Logic Programming. In D.H.D. Warren and P. Szeredi, editors, *Proc. 7th Int. Conference on Logic Programming*, pages 481–494. The MIT Press, 1990.

[30] R. A. O'Keefe. Towards an algebra for constructing logic programs. In *IEEE Symposium on Logic Programming*, 1985.

[31] Teodor Przymusinski. Perfect Model Semantics. In R. A. Kowalski and K. A. Bowen, editors, *Proc. 5th Int. Conference on Logic Programming*, pages 1081–1096. The MIT Press, 1988.

[32] Prasad Rao, Konstantinos F. Sagonas, Terrance Swift, David Scott Warren, and Juliana Freire. Xsb: A system for effciently computing wfs. In *Proceedings of the Intl. Conference on Logic Programming and Non Monotonic Reasonning*, pages 431–441, 1997.

[33] Konstantinos F. Sagonas, Terrance Swift, and David Scott Warren. Xsb as an efficient deductive database engine. In Richard T. Snodgrass and Marianne Winslett, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota*, pages 442–453. ACM Press, 1994.

[34] S. Shaw, L. Foggiato-Bish, I. Garcia, G. Tillman, D. Tryon, W. Wood, and C. Zaniolo. Improving data quality via $\mathcal{LDL}++$. In *Workshop on Deductive Databases, JICSLP*, pages 60–73, 1993.

[35] D. S. Touretzky. *The Mathematics of Inheritance Systems*. Morgan Kaufmann, Los Altos, CA, 1986.

[36] J. D. Ullman. *Principles of Database and Knowledge-base Systems*. Computer Science Press, 1988.

[37] M. H. van Emden. Quantitative deduction and its fixpoint theory. *Journal of Logic Programming*, 4(1):37–53, 1986.

[38] J. H. You, S. Ghosh, L. Y. Yuan, and R. Goebel. An introspective framework for paraconsistent logic programs. In John Lloyd, editor, *Proceedings of the 12th International Logic Programming Symposium*, pages 384–398, Portland, Oregon, December 1995. MIT Press.

[39] C. Zaniolo. Object-oriented programming in Prolog. In *International Symposium on Logic Programming*, pages 265–270, Atlantic City, February 1984.

[40] C. Zaniolo. Object identity and inheritance in deductive databases – an evolutionary approach. In W. Kim, J.-M. Nicolas, and S. Nishio, editors, *Proc. of the 1st DOOD Conference*, pages 7–24, Kyoto, Japan, 1990. North-Holland.

Nonmonotonic Reasoning II

In search of the right extension

Jérôme Lang
IRIT-UPS
31062 Toulouse Cedex
France
lang@irit.fr

Pierre Marquis
CRIL/Université d'Artois
62307 Lens Cedex
France
marquis@cril.univ-artois.fr

Abstract

In this paper, the problem of purifying an assumption-based theory KB, i.e., identifying the right extension of KB using knowledge-gathering actions (tests), is addressed. Assumptions are just normal defaults without prerequisite. Each assumption represents all the information conveyed by an agent, and every agent is associated with a (possibly empty) set of tests. Through the execution of tests, the epistemic status of assumptions can change from "plausible" to "certainly true", "certainly false" or "irrelevant", and the KB must be revised so as to incorporate such a change. Because removing all the extensions of an assumption-based theory except one enables both identifying a larger set of plausible pieces of information and renders inference computationally easier, we are specifically interested in finding out sets of tests allowing to purify a KB (whatever their outcomes). We address this problem especially from the point of view of computational complexity.

1 Introduction

In the following, knowledge bases given by propositional assumption-based theories $KB = \langle W, \Delta \rangle$ (or equivalently, prerequisite-free normal default theories), are considered. W is a finite set of propositional formulas representing certain information about *the real world*, the latter being characterized by a specific (a priori unknown) model I. Δ is a finite set of defaults (or assumptions); defaults are just propositional formulas that are plausible, i.e., considered as true in absence of any contradictory information. Since defaults may be jointly contradictory, several belief states (called extensions) may result. Equivalently, we can also consider KB as a (typically inconsistent) stratified belief base, with two strata (the first one W being consistent).

The purpose of reasoning from KB is to identify formulas that are true in I, ideally all of them. Of course, in the general case, only some of the formulas true in I are accessible. Anyway, an important question is: what are the logical links between what is derivable from KB and what is true in I? When dealing with assumption-based theories, the following representation (meta)assumption is typically made:

(D) There exists a extension of KB (noted E_I) that is true in I.

Note that E_I is necessarily unique (because any two extensions of KB are jointly contradictory). It is important to note that (D) is a representation assumption (just like the one consisting in assuming that W is true in I). Eventually, it can be defeated at the light of certain information. Thus, there is no guarantee that every formula derived from KB through skeptical inference (i.e., every formula belonging to all the extensions of KB) is true in I. Nevertheless, such a (meta)assumption is mandatory to take defaults into account in reasoning. Without it, only the logical consequences of W would be considered as true in I and pure deduction from W is definitely an over-cautious form of inference in presence of defaults (since it does not exploit them at all).

Now, what can be done if we want to identify some other formulas as true in I? A classical way ([2]) consists in taking into consideration representation (meta)assumptions that are stronger than (D). Especially, if some priorities on defaults are available (e.g., a pre-order over Δ reflecting the fact that some defaults are more plausible than others is given), it can be valuable to take it into account and to consider that at least one *preferred* extension of KB is true in I. Indeed, restricting the set of extensions of KB by focus-

ing on the preferred ones may only lead to enlarge their intersection (hence the set of formulas that are considered as true in I). An extreme, most informative, case is achieved when all the extensions of KB are filtered out, except one. In this situation, every formula from the remaining extension is considered true in I. Additionally, provided that KB has only one extension, assumption-based reasoning becomes computationally easier (from the second level of the polynomial hierarchy to the first level)[1]. This is another reason that makes it highly desirable to have KB with one extension.

In this paper, a radically different but complementary approach is developed. We assume that every default from Δ represents all the pieces of information conveyed by one agent (a source of information); W represents the set of all pieces of information on which all agents agree (e.g., what must be true by nature). We also assume that each agent can perform some knowledge-gathering actions (informative tests). At the light of tests outcomes, the group of agents can change its mind about the epistemic status of defaults. As a consequence, revising KB with the outcome of a test can easily modify the set of its extensions. Specifically, contrariwise to what happens in the preferred extension approach, revising KB can lead to remove defaults in some extensions or even to generate new extensions, thus questioning the (meta)assumption (D).

In the following, each agent is associated with a single test whose outcome directly validates the corresponding default (changing its status from "plausible" to "certainly true"), invalidates the default (changing its status from "plausible" to "certainly false") or cancels it (changing its status to "irrelevant"). We show how such default verification tests can be generated from more general tests through a pre-processing phase, and we analyze the complexity of the pre-processing. *Purifying* a KB consists in determining a set of tests whose outcomes (whatever they are) generate an assumption-based theory with a unique extension after revision. We analyze the purification problem from the computational complexity point of view.

[1]When KB has only one extension, skeptical inference from KB (as well as argumentative and brave inference since, in this case, they all coincide) becomes computationally easier; indeed, while skeptical inference is Π_2^p-complete in the general case [7] [21], the complexity falls down to coNP-complete, when KB has a unique extension E_S (once its generating scenario S has been found out – and this can be achieved using only $card(\Delta)$ calls to an NP oracle).

2 Basics of assumption-based reasoning

We assume the reader acquainted with basic notions of propositional logic. Let $PROP_{PS}$ be a propositional language built up in the usual way from a set of variables PS. Throughout this paper, knowledge bases are represented by propositional assumption-based theories [16].

Definition 1 (assumption-based reasoning)

- *An* assumption-based theory KB *is an ordered pair* $KB = \langle W, \Delta \rangle$, *where* W *is a (conjunctively-interpreted) finite subset of* $PROP_{PS}$, *and* $\Delta = \{\phi_1, \ldots, \phi_n\}$ *is a finite subset of* $PROP_{PS}$ [2].
- *A* nogood *of* KB *is a minimal subset* N *of* Δ *(w.r.t.* $\subseteq$*) such that* $N \cup W$ *is not satisfiable.* $Nogood(KB)$ *is the set of all nogoods of* KB.
- *A* maximal scenario *of* KB *is a maximal subset* S *of* Δ *(w.r.t.* $\subseteq$*) such that* $S \cup W$ *is satisfiable.* $MaxScen(KB)$ *is the set of all maximal scenarios of* KB.
- *An* extension E_S *of* KB *is the deductive closure* $Cn(S \cup W)$, *where* S *is a maximal scenario of* KB. S *is called the* generating scenario *of* E_S. $Ext(KB)$ *is the set of all extensions of* KB.

Example 1 *Let* $KB = \langle W, \Delta \rangle$ *with* $W = \emptyset$ *and* $\Delta = \{a \wedge b, \neg a, \neg b \wedge c, d\}$.
We have $Nogood(KB) = \{\{a \wedge b, \neg a\}, \{a \wedge b, \neg b \wedge c\}\}$, $MaxScen(KB) = \{\{a \wedge b, d\}, \{\neg a, \neg b \wedge c, d\}\}$, *and* $Ext(KB) = \{Cn(\{a \wedge b, d\}), Cn(\{\neg a, \neg b \wedge c, d\})\}$.

Every assumption-based theory KB has at least a maximal scenario whenever W is satisfiable, and more than one whenever it has a nogood containing more than one assumption. It is easy to see that $MaxScen(KB)$ and $Ext(KB)$ are two similar sets since every extension of KB is generated by a unique maximal scenario of KB (the maximality requirement ensures that the union of two distinct maximal scenarios of KB is unsatisfiable with W). Accordingly, determining whether KB has only one extension can be decided in linear time when either $Nogood(KB)$ or $MaxScen(KB)$ are given (but both sets may contain exponentially many elements w.r.t. the size of KB in

[2]At a first glance, considering Δ as a set can appear too restrictive since it can be the case that two different agents convey the same piece of information: hence, a bag (or multiset) would be more convenient. Nevertheless, for simplicity and without loss of generality, Δ will be considered as a set: if two identical defaults are to be considered, it is sufficient to make them syntactically different, for instance by indexing them with integers, while preserving equivalence.

the worst case). It is well-known that $Nogood(KB)$ can be generated from $MaxScen(KB)$ and *vice-versa*, through the computation of minimal hitting sets [18] and set-theoretic complements in Δ.

3 Revising defaults using tests

In the following, we assume that each default ϕ_i of Δ represents all the information conveyed by the agent (or source) i. Some of the defaults ϕ_i are associated with a test t_i (at most one per default). Executing t_i can change the epistemic status of ϕ_i (considered plausible before execution) in one of the following ways:

- ϕ_i becomes certainly true, noted $ok(i)$.
- ϕ_i becomes certainly false, noted $bad(i)$.
- ϕ_i becomes irrelevant, noted $forget(i)$.

Definition 2 (default verification test) *A* default verification test t_i *consists of a subset of* $\{ok(i), bad(i), forget(i)\}$ *containing* $ok(i)$ *and at least one additional element. The elements* r_i *of* t_i *are its* possible outcomes. *To each test* t_i *is attached a positive integer* k_i *representing the* cost *of performing* t_i.

Since the very purpose of testing a default ϕ_i is to validate it, $ok(i)$ is required to be among the possible outcomes of it.

While such tests can be considered quite restricted at a first glance, they nevertheless capture several important notions of verification:

- *default verification*: the piece of information itself (ϕ_i) given by the source i is verified. Assume for example that undergrad student Tweety programmed an algorithm for testing primality, which resulted after some laborious hours in software S. Running S on number 348754697 returns "yes" (ϕ_i is thus "348754697 is prime"). Tweety's ability in programming is known to be average, therefore, ϕ_i should be considered a default and not a fact. Now, Lea is an expert in prime numbers. She are fully reliable, thus, her answer to the question whether a given number is prime is always correct. Furthermore, Lea always gives an answer whenever the input number does not exceed a given maximum number N_{max} (much larger than 348754697), and sometimes gives an answer when the input number exceeds N_{max}. Asking Lea about the primality of 348754697 is a *default verification test* of the form $\{ok(i), bad(i)\}$: its outcome is $ok(i)$ (resp. $bad(i)$) if Lea answers "yes" (resp. "no"). Some years later, Tweety improves his software so as to deal with very large integers. Running the improved software on $N = 495729^{50876} + 34532^{3486}$ (which exceeds N_{max}) returns "yes", which leads to considering the default ϕ_j "N is prime". Asking Lea about the primality of N is a default verification test of the form $\{ok(j), bad(j), forget(j)\}$, because $N > N_{max}$. If Lea is not able to answer, i.e., if the outcome of the test is $forget(j)$, then we will not be ready to consider N as prime any longer, because Lea is an expert and the fact that she was not able to answer is a reason strong enough for changing the status of ϕ_j from plausible to irrelevant.

- *source verification*: the reliability of the source, thus the *relevance* of the piece of information, is checked. Not the piece of information itself. Bob is Tweety's teacher. Asking Bob to look carefully on Tweety's first program S returns yes or no whether S is valid or not. If it is, then the information S gave was correct and 348754697 is prime. If it is not, then anything could happen. The program might have been something like generating "yes" or "no" randomly. This cancels the default information that ϕ_i is true but certainly does not tell us that ϕ_i is false. It could well be true in I, *by chance* (here, 348754697 may well be prime even if S is not valid). Asking Bob is a test of the form $\{ok(i), forget(i)\}$.

Now, once t_i has been executed, KB must be revised so as to incorporate the information given by its resulting outcome r_i into KB. The revision process aims at characterizing the new assumption-based theory $KB * r_i$ obtained by incorporating the information given by r_i within KB:

Definition 3 (revising KB with a test outcome) *Let* $KB = \langle W, \Delta \rangle$ *an assumption-based theory, and* t_i *be a test concerning default* ϕ_i *of* Δ. *We have:*

- **default validation**

$$KB * ok(i) = \langle W \cup \{\phi_i\}, \Delta \setminus \{\phi_i\} \rangle$$

The execution of t_i *has validated* ϕ_i. *The default is definitely true in* I. *It can be soundly added to* W *and removed from* Δ[3].

- **default invalidation**

$$KB * bad(i) = \langle W \cup \{\neg\phi_i\}, \Delta \setminus \{\phi_i\} \rangle$$

[3]Note that letting ϕ_i in Δ or not once ϕ_i has been added to W has no influence on $Ext(KB * ok(i))$.

The execution of t_i has invalidated ϕ_i. The default is definitely false in I. Its negation can be soundly added to W and ϕ_i can be removed from Δ [4].

- **default cancellation**

$$KB * forget(i) = \langle W, \Delta \setminus \{\phi_i\}\rangle$$

The default is considered irrelevant, it is removed from Δ. A frequent reason for cancelling a default is a source verification that fails (as explained above); the failure can be viewed as an argument against the default, but of course, this argument is not sufficiently strong for invalidating the default (we just consider it as irrelevant).

Tests are assumed reliable (their outcome is true in I) and non-intrusive (they have no effect on the world). Accordingly, revising KB by a sequence of test outcomes depends only on the outcome of each individual test, and is independent from the ordering in which tests are performed.

Definition 4 (verification problems)

- *A* verification problem *is an ordered pair $P = \langle KB, T\rangle$, where $KB = \langle W, \Delta\rangle$ is an assumption-based theory and $T = \{t_1, \ldots, t_k\}$ is the available set of tests. For every $\phi_i \in \Delta$, ϕ_i is* testable *iff $t_i \in T$; without loss of generality, we assume that only the first k defaults $\phi_1 \ldots \phi_k$ of Δ are testable ($0 \leq k \leq card(\Delta)$).*
- *A* set of tests *$X = \{t_1, \ldots t_l\}$ (given P) is any subset of T.*
- *Executing a set of tests $X = \{t_1, \ldots t_l\}$ results in a* state *(for P), i.e., a set $R = \{r_1, \ldots, r_l\}$ where each r_i ($i \in 1 \,..\, l$) is an element of t_i (one of its possible outcomes). It is convenient to partition every state R in $\{Ok(R), Bad(R), Forget(R)\}$ with $Ok(R) = \{i \mid ok(i) \in R\}$, $Bad(R) = \{i \mid bad(i) \in R\}$, $Forget(R) = \{i \mid forget(i) \in R\}$. $State(X)$ is the set of states that can possibly result from the execution of X.*
- *A state R (for P) is* consistent *iff $W \cup \{\phi_i \mid i \in Ok(R)\} \cup \{\neg\phi_i \mid i \in Bad(R)\}$ is satisfiable. Clearly, since tests are assumed reliable, all states in $State(X)$ are consistent.*
- *The* revision *of KB by state $R = \{r_1, \ldots, r_l\}$, noted $KB * R$, is the assumption-based theory defined by $KB * r_1 * \ldots * r_l$.*

[4] Note that letting ϕ_i in Δ or not once $\neg\phi_i$ has been added to W has no influence on $Ext(KB * bad(i))$.

For our purpose, it is important to identify the effect of each possible default revision (validation, invalidation, cancellation) on $Ext(KB)$. Since it is possible to determine efficiently whether KB has a unique extension given $Nogood(KB)$ or $MaxScen(KB)$, it is also important to analyze how both sets evolve after a revision.

The case of default validation corresponds to a "soft" knowledge expansion. By "soft" we mean that there is no *surprise*: what was considered as a plausible piece of information becomes now a certain piece of information. The effects are a (possible) removing of some of the current extensions, but never the generation of a new extension (point 3 below).

We make use of the following notations: if A is a set of subsets of a given set, then
$A_{min\subset} = \{B \in A \mid \text{there is no } B' \in A \text{ s.t. } B' \subset B\}$ and
$A_{min\supset} = \{B \in A \mid \text{there is no } B' \in A \text{ s.t. } B' \supset B\}$.

Proposition 1 (effects of a default validation)

1. $Nogood(KB * ok(i)) = \{N \setminus \{\phi_i\} \mid N \in Nogood(KB)\}_{min\subset}$.
2. $MaxScen(KB * ok(i)) = \{S \setminus \{\phi_i\} \mid S \in MaxScen(KB) \text{ and } \phi_i \in S\}$.
3. $Ext(KB * ok(i)) \subseteq Ext(KB)$.

Thus, the extensions of $KB * ok(i)$ are exactly those extensions of $Ext(KB)$ which contain ϕ_i. The other ones are removed. Consider Example 1 again with $\phi_1 = a \wedge b$. We have $Nogood(KB * ok(1)) = \{\{\neg a\}, \{\neg b \wedge c\}\}$ and $MaxScen(KB * ok(1)) = \{\{d\}\}$. Accordingly, $KB * ok(1)$ has a unique extension $Cn(\{a \wedge b, d\})$. Clearly enough, point 3. can be easily generalized (by induction) to *sequences of validation*: namely, if a state R contains only $ok(i)$ variables, then $Ext(KB * R) \subseteq Ext(KB)$.

Proposition 2 (effects of a default cancellation)

1. $Nogood(KB * forget(i)) = \{N \in Nogood(KB) \mid \phi_i \notin N\}$.
2. $MaxScen(KB * forget(i)) = \{S \setminus \{\phi_i\} \mid S \in MaxScen(KB)\}_{min\supset}$.
3. $\forall E \in Ext(KB * forget(i))$, *there is a* $E' \in Ext(KB)$ *such that* $E \subseteq E'$.

Default cancellation can be considered as a simple form of belief contraction. However, a default ϕ_i can survive its contraction in some sense; indeed, it can be the case that ϕ_i is a (skeptical) consequence of $KB * forget(i)$ (just consider the case where $\Delta = \{\phi_i, \phi_j\}$, $W \cup \{\phi_j\}$ is satisfiable and $\phi_j \models \phi_i$ holds). This is not an undesirable effect: if ϕ_i is a logical consequence of W

when some defaults of Δ are validated (or only considered as true), the fact that the corresponding verification test t_i has failed is not a sufficient argument for considering ϕ_i as irrelevant. The discrepancy with the belief contraction situation can be explained by the fact that, while both cancellation and contraction change the status of some information to "irrelevant", the status of a default before change is "plausible", not "certainly true".

With default cancellation, preservation of extensions is not guaranteed; indeed, it may be the case that one of the extensions of KB leads to a weaker one. Consider Example 1 again, where $\phi_2 = \neg a$. We have $Nogood(KB * forget(2)) = \{\{a \wedge b, \neg b \wedge c\}\}$ and $MaxScen(KB * forget(2)) = \{\{a \wedge b, d\}, \{\neg b \wedge c, d\}\}$. Thus $KB * forget(2)$ has two extensions, $E_1 = Cn(\{a \wedge b, d\})$ and $E_2 = Cn(\{\neg b \wedge c, d\})$. While E_1 is an extension of KB, E_2 is only a proper subset of the extension $Cn(\{\neg a, \neg b \wedge c, d\})$ of KB.

When it comes to sequential revision, generalizing points 3. of both Propositions 1 and 2 by induction on $card(R)$ easily gives the following result: if a state R is "bad-free", i.e., $Bad(R) = \emptyset$, then $\forall E \in Ext(KB * R)$, there is a $E' \in Ext(KB)$ s.t. $E \subseteq E'$. As a corollary, we have that $card(Ext(KB * R)) \leq card(Ext(KB))$: revising a KB with a "bad-free" state R does not increase the *number* of extensions of KB.

Default invalidation can be considered as a knowledge expansion with the negation of the default. Nevertheless, it is trickier than default validation: there may be new extensions appearing, and the number of extensions may increase. These more chaotic effects are in agreement with the intuition which tells that a default invalidation change the status of "plausible" to "certainly false", which leads to a *surprise* (contrariwise to default validation). Consider Example 1 again. We have $Nogood(KB * bad(2)) = \{\{a \wedge b, \neg b \wedge c\}\}$ and $MaxScen(KB * bad(2)) = \{\{a \wedge b, d\}, \{\neg b \wedge c, d\}\}$, thus $KB * bad(2)$ has two extensions: E_1 as above, and $E_3 = Cn(\{a, \neg b \wedge c, d\})$ *which is a new extension* – incompatible with any extension of KB.

Furthermore, while Propositions 1 and 2 show how both the nogoods and the maximal scenarios of a revised KB can be computed (in polynomial time) from, respectively, the nogoods and the maximal scenarios of KB whenever the revision operation is a default validation or a default cancellation, no similar results can be achieved when considering a default invalidation. Indeed, there is not enough information in $Nogood(KB)$ (or $MaxScen(KB)$) to compute $Nogood(KB * bad(i))$ (or $MaxScen(KB * bad(i))$) in the general case:

Example 2 *Let* $KB = \langle W, \Delta \rangle$, *with* $W = \{a \vee \neg b \vee \neg c\}$ *and* $\Delta = \{a, b, c\}$. *We have* $Ext(KB) = \{Cn(W \cup \{a, b, c\})\}$, $MaxScen(KB) = \{\{a, b, c\}\}$, *and* $Nogood(KB) = \emptyset$. *Now, given that* $\phi_1 = a$, $Ext(KB * bad(1)) = \{Cn(W \cup \{\neg a\} \cup \{b\}), Cn(W \cup \{\neg a\} \cup \{c\})\}$, $MaxScen(KB * bad(1)) = \{\{b\}, \{c\}\}$, *and* $Nogood(KB * bad(1)) = \{\{b, c\}\}$. *Let* $KB' = \langle \emptyset, \Delta \rangle$. *We have* $MaxScen(KB') = MaxScen(KB)$ *and* $Nogood(KB') = Nogood(KB)$. *Nevertheless, we do* **not** *have* $Ext(KB' * bad(1)) = Ext(KB * bad(1))$ *since* $Ext(KB' * bad(1)) = \{Cn(\{\neg a, b, c\})\}$.

Accordingly, while knowing only nogoods or maximal scenarios is sufficient to determine whether KB, or the revision of KB by a sequence of validations and cancellations, has only one extension, this is no longer the case when default invalidations are possible. This calls for a more informative notion, like the one of *generalized nogood*:

Definition 5 (generalized nogood)
A generalized nogood *of* KB *is a minimal (w.r.t.* $\subseteq$*) subset* G *of* $\Delta \cup \{\neg\phi_i \mid \phi_i \in \Delta\}$ *such that* $G \cup W$ *is not satisfiable.* $GNogood(KB)$ *is the set of all generalized nogoods of* KB.

Clearly enough, we have
$GNogood(\langle W, \Delta \rangle) = Nogood(\langle W, \Delta \cup \{\neg\phi_i \mid \phi_i \in \Delta\}\rangle)$
and
$Nogood(\langle W, \Delta \rangle) = \{G \in GNogood(\langle W, \Delta \rangle) \mid G \subseteq \Delta\}$.
Accordingly, $Nogood(KB)$ can be computed in polynomial time from $GNogood(KB)$ by removing from it every element that is not a subset of Δ. Additionally:

Proposition 3 (computing $GNogood(KB * r_i)$)

1. $GNogood(KB * ok(i)) = \{G \setminus \{\phi_i\} | G \in GNogood(KB)\}_{minC}$.
2. $GNogood(KB * forget(i)) = \{G \in GNogood(KB) | \phi_i \notin G\}$.
3. $GNogood(KB * bad(i)) = \{G \setminus \{\neg\phi_i\} | G \in GNogood(KB)\}_{minC}$.

Accordingly, computing $GNogood(KB * R)$ from $GNogood(KB)$ and a consistent state R can be achieved in polynomial time.

4 Purifying KB

In this section, we focus on the problem consisting of the identification of *the* right extension E_I of KB. To this end, we look for sets of tests which discriminate among all extensions, so as to find E_I. This calls for the notion of *purification* of KB by a set of tests X.

4.1 N-purification vs. C-purification

As seen before, revising KB with a state may eventually generate extensions that are not in $Ext(KB)$, hence questioning the (meta)assumption (D). As a consequence, two different notions of purification must be considered, related to the fact that we are ready to defeat (D) or not. If (D) can be relaxed at the light of a new evidence, the objective of the revision process is that the revised KB has only one extension. If (D) cannot be questioned, a valuable revision process can lead to a revised KB with more than one extension, but only one of them must be among the initial extensions of KB.

Definition 6 (purifying states)
Given a verification problem $P = \langle KB, T\rangle$ and a consistent state R:

- R N-purifies KB *iff* $KB * R$ *has only one extension.*
- R C-purifies KB *iff* $KB * R$ *has only one extension E s.t. $E \in Ext(KB)$.*

Certainly, the uniqueness requirement in both definitions is quite idealistic and in practice one would rather try to do as good as possible (identifying as many inconsistencies as possible, even if it cannot be done totally). For the sake of simplicity, we ignore this issue in this paper.

It is important to see why both definitions of purification do not coincide (and none of them implies or excludes the other one):

Example 3 *Let $KB = \langle \emptyset, \{\neg a \wedge \neg b, a, b\}\rangle$. We have $Nogood(KB) = \{\{\neg a \wedge \neg b, a\}, \{\neg a \wedge \neg b, b\}\}$ and $MaxScen(KB) = \{\{\neg a \wedge \neg b\}, \{a, b\}\}$; thus, KB has two extensions, namely $E = Cn(\{\neg a \wedge \neg b\})$ and $E' = Cn(\{a, b\})$. Let $\phi_1 = \neg a \wedge \neg b$, $\phi_2 = a$, and $\phi_3 = b$. Let $R = \{bad(2)\}$. $KB * R = \langle\{\neg a\}, \{\neg a \wedge \neg b, b\}\rangle$ has two extensions, namely $Cn(\{\neg a \wedge \neg b\})$ and $Cn(\{\neg a, b\})$, and therefore R does not N-purify KB. Contrastingly, since $Ext(KB) \cap Ext(KB * R) = \{Cn(\{\neg a \wedge \neg b\})\}$ is a singleton, R C-purifies KB. Now, let $R' = \{forget(1), bad(2)\}$; $KB * R' = \langle\{\neg a\}, \{b\}\rangle$ has only one extension $Cn(\{\neg a, b\})$ and thus R' N-purifies KB, while R' does not C-purify KB. Lastly, $R'' = \{ok(3)\}$ both N-purifies and C-purifies KB.*

In N- and C-purification, N and C stand respectively for *no completion* and *completion*, which is justified by the following result where "forget-free" states are considered, only:

Proposition 4 (C- and N-purification)
Let $P = \langle KB, T\rangle$ be a verification problem, with $KB = \langle W, \Delta\rangle$, and let R be a consistent "forget-free" state (i.e., $Forget(R) = \emptyset$). The two statements below are equivalent:

1. *R C-purifies KB.*
2. *R N-purifies $\langle Compl_\Delta(W), \Delta\rangle$, where $Compl_\Delta(W) = W \cup \{\bigvee \{S \mid S \in MaxScen(KB)\}\}$ is the* completion *of W w.r.t. Δ.*

Clearly enough, replacing W by $Compl_\Delta(W)$ in KB amounts to considering that (D) cannot be questioned; indeed, the deductive closure of $Compl_\Delta(W)$ is equal to the deductive closure of W interpreted under (D).

Which one of both definitions should be used in practice, i.e., whether we are ready to defeat (D) or not, is a nontrivial problem (which is to be considered as well in model-based diagnosis). Stated otherwise, this question becomes: when taking account of an observation for discriminating among extensions (or diagnoses), should we start from the initial set of extensions and filter out those that are incompatible with the observation (C-purification) or expand the certain knowledge with the observation and then recomputing the extensions (N-purification)?

4.2 Computing purifying sets

In the following, we focus on N-purification, only. We are interested in determining sets of tests that enable purifying KB whatever their outcomes. We are specifically interested in the minimal ones w.r.t. $\subseteq$, called purification bases.

Definition 7 (purifying sets of tests, bases)
Given a verification problem $P = \langle KB, T\rangle$, and a set $X \subseteq T$ of tests:

- X N-purifies KB *iff for any consistent state $R \in State(X)$, R N-purifies KB. X is said a* purifying set *for KB.*
- *X is a* N-purification base *(or base for short) for KB iff X N-purifies KB and no proper subset of it does it. $Base(KB)$ is the set of all N-purification bases for KB.*
- *KB is* N-purifiable *iff there exists a N-purification base (or, equivalently, if there exists a purifying set) for it.*

In order to figure out the complexity of computing purifying sets, we investigate the complexity of the two following decision problems (we assume the reader familiar with complexity theory, in particular with the polynomial hierarchy (see for instance [15]):

- PURIFYING STATE: does a given state R N-purify KB?
- PURIFYING SET: does a given set X of tests N-purify KB?

Proposition 5 (PURIFYING STATE)
PURIFYING STATE *is in* $\Delta_2^p[\mathcal{O}(n)]$, *and is both* NP*-hard and* coNP*-hard.*

Accordingly, checking whether a given assumption-based theory has a unique extension is not in NP ∪ coNP, unless NP = coNP (which is considered very unlikely).

Proposition 6 (PURIFYING SET)
PURIFYING SET *is* Π_2^p*-complete.*

These complexity results show the computation of purifying sets highly intractable. Indeed, checking whether a given set X is a purifying set for KB is already Π_2^p-hard. This is why it is important to identify restrictions for which the computational task can prove easier.

Accordingly, in the following, we assume that the available set T of tests contains only source verification tests, i.e., tests t_i of the form $\{ok(i), forget(i)\}$. Stated otherwise, we assume that performing any set X of tests can only lead to "bad-free" states. This assumption is called (`bad-free`) in the following. A common situation where the (`bad-free`) assumption holds is when the only test considered are *source verification tests.*

Interestingly, purifying sets for KB can be characterized from $Nogood(KB)$ under this assumption:

Proposition 7 (purification under (`bad-free`))
Let $P = \langle KB, T\rangle$ a verification problem, where T consists of source verification tests, only. For any nogood N of KB, let $Index(N) = \{i \mid \phi_i \in N\}$.

1. *A consistent state R N-purifies KB if and only if for every nogood N of KB, at least one of the following three statements is true:*
   - *(i)* $Index(N) \cap Forget(R) \neq \emptyset$;
   - *(ii)* $Index(N) \setminus Ok(R)$ *is a singleton;*
   - *(iii)* $\exists N' \in Nogood(KB)$ *s.t.* $Index(N') \setminus Ok(R)$ *is a singleton* $\{i\}$ *with* $i \in Index(N)$.
2. *A set X of tests N-purifies KB if and only if for every nogood N of KB, we have* $card(Index(N) \setminus \{i | t_i \in X\}) \leq 1$.

The first point means that R purifies KB if (i) every nogood N is broken by the cancellation of a default $\phi_i \in N$, or if (ii) all defaults of N except one have been validated, which enables to deduce that the remaining one has to be cancelled, or (iii) if it has been possible to deduce by reasoning as in (ii) about another nogood N' that a given default ϕ_i of N has to be cancelled. The second point means that X purifies KB if and only if every nogood contains at most a default whose associated test t_i is not in X.

The above proposition shows that the complexity of both PURIFYING STATE and PURIFYING SET decreases to P when the input is $Nogood(KB)$ and X. As an additional corollary, we have:

Corollary 1 *Let $P = \langle KB, T\rangle$ a verification problem, where T consists of source verification tests, only.*

- *If X and X' are two sets of tests such that $X \subseteq X'$ and X N-purifies KB, then X' N-purifies KB as well.*
- *KB is N-purifiable iff T N-purifies KB.*

Clearly enough, this corollary would not hold in the general case, if the (`bad-free`) assumption were not made (see Example 3 with $X = \emptyset$ and $X' = \{\{ok(1), bad(1)\}\}$ for a counterexample).

Therefore, while the (`bad-free`) assumption does not lower the complexity of PURIFYING SET in the general case, it enables determining whether a KB is N-purifiable in polynomial time whenever the input is $Nogood(KB)$ and T. Moreover, in the case where KB is N-purifiable, a base for KB can be computed in polynomial time from $Nogood(KB)$ and T thanks to the following greedy algorithm: start with $B = T$, for every element t of B, remove t from B if $B \setminus \{t\}$ still N-purifies KB. Once all the tests of T have been processed, the resulting set B is, by construction, a base for KB. This algorithm can also be used to determine in polynomial time whether or not a given set X of tests is a base for KB (just start the algorithm with $B = X$: if the resulting set of tests is X, then X is a base for KB, otherwise it is not). Accordingly, the computation of $Base(KB)$ can be achieved through a simple generate-and-test procedure using the previous greedy algorithm as a routine[5].

Example 4 *Let $P = \langle\langle W, \Delta\rangle, T\rangle$ a verification problem, where $\Delta = \{\phi_1, \ldots, \phi_7\}$, T consists of source verification tests only, and each default ϕ_i ($i \in 1...7$) is testable. Assume that*

$$Nogood(KB) = \{ \{\phi_1, \phi_2, \phi_5\}, \{\phi_2, \phi_6\}, \{\phi_3, \phi_6\}, \{\phi_4, \phi_5, \phi_6\}, \{\phi_7\}\}.$$

[5]The search can be organized using the SE-tree framework given in [19].

We have

$$Base(KB) = \{ \quad \{t_1,t_2,t_4,t_6\}, \{t_1,t_3,t_4,t_6\}, \{t_1,t_5,t_6\}, \{t_2,t_3,t_4,t_5\}, \{t_2,t_5,t_6\}\}.$$

The tractability of PURIFYING SET suggests that the search for *preferred bases*, i.e., bases of minimal cost, could be envisioned under the (**bad-free**) assumption when the input is $Nogood(KB)$ and T. Here, several aggregation policies can be considered in order to define the *cost* $k(X)$ of a set X of tests:

- $k(X) = \sum_{i \in X} k_i$ (SUMCOST policy);
- $k(X) = \max_{i \in X} k_i$ (MAXCOST policy).

The MAXCOST policy is particularly well-suited to the situation where the individual costs k_i represent running times and all tests are performed in parallel (in this case, the overall running time corresponds to the duration of the longest task). The SUMCOST policy must be considered in many of the remaining cases (especially, when the k_i represent monetary costs, or running times when the verification tasks are performed in a pure sequential way).

We have identified the following complexity results:

Proposition 8 (MINIMAL COST BASE)
Let $P = \langle KB, T\rangle$ a verification problem, where T consists of source verification tests, only. Given $Nogood(KB)$, T and an integer α, determining whether there exists a base $X \subseteq T$ for KB such that $k(X) \leq \alpha$ is in P *under the* MAXCOST *policy and is* NP*-complete under the* SUMCOST *policy.*

Let us now show how minimal cost bases for KB can be computed.

Under the MAXCOST policy.

Clearly enough, there exists a base X for KB s.t. $k(X) \leq \alpha$ iff $\{t_i \in T | k_i \leq \alpha\}$ N-purifies KB, which can be tested in polynomial time. Interestingly, it is possible to derive in polynomial time the smallest α such that $\{t_i \in T | k_i \leq \alpha\}$ N-purifies KB. We note k^* this smallest α. To determine the value of k^*, let us consider the mapping **Secondmax** that associates to every finite multi-set of positive integers the second element of the multi-set sorted in decreasing order, and 0 if the multi-set contains less than two elements. For instance, **Secondmax**$(\{4,6,6,3,9,5,5\}) = 6$, **Secondmax**$(\{4,7,1,7\}) = 7$, **Secondmax**$(\{4\}) = 0$. We have:

Proposition 9 (computing k^*)
$k^* = max_{N \in Nogood(KB)}$ **Secondmax**$(\{k_i\}_{\phi_i \in N})$.

As a direct corollary, we have:

Corollary 2 *$\{t_i \in T \mid k_i \leq k^*\}$ is a purifying set for KB that is of minimal cost under the* MAXCOST *policy.*

Such a set is not necessarily a minimal cost base for KB since it is not necessarily minimal w.r.t. $\subseteq$. Nevertheless, applying the greedy algorithm sketched above to this set enables deriving in polynomial time a minimal cost base for KB.

Example 4 (continued) Assuming that each test t_i has cost i, we have $k^* = 5$, hence $\{t_i \in T \mid k_i \leq k^*\} = \{t_1,t_2,t_3,t_4,t_5\}$ is a purifying set for KB of minimal cost, and the minimal cost base $\{t_2,t_3,t_4,t_5\}$ for it can be derived from the latter set in polynomial time.

Under the SUMCOST policy.

Since the decision problem associated to the optimisation problem we are interested in is NP-complete, there is little hope that a (deterministic) polynomial time algorithm for computing a minimal cost base for KB exists, under the SUMCOST policy. Nevertheless, this does not prevent us from designing algorithms that can exhibit a satisfying computational behaviour for many instances. Interestingly enough, as the next proposition shows it, we can take advantage of numerous algorithms for $0-1$ integer programming in order to achieve our goal:

Proposition 10 (0-1 linear programming)
Let $P = \langle KB, T\rangle$ a verification problem, where T consists of source verification tests, only. $X \subseteq T$ is a minimal cost base for KB under the SUMCOST *policy iff the vector $(x_1, ..., x_n)$ such that $x_i = 1$ if $t_i \in X$ and $x_i = 0$ otherwise, is a solution of the following* 0-1 *linear program:*

$$min(\sum_{i=1}^{n} k_i x_i)$$

$$s.t. \begin{cases} \forall i = 1...n, x_i \in \{0,1\} \\ \forall N \in Nogood(KB), \sum_{\phi_i \in N} x_i \geq card(N) - 1. \end{cases}$$

Example 4 (continued) The $0-1$ linear program associated to the previous example when every default has a unary cost is:

$$min(x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7)$$

$$\text{s.t.} \begin{cases} x_1 + x_2 + x_5 \geq 2 \\ x_2 + x_6 \geq 1 \\ x_3 + x_6 \geq 1 \\ x_4 + x_5 + x_6 \geq 2 \\ x_7 \geq 0 \end{cases}$$

The following vectors are optimal solutions: $(1,0,0,0,1,1,0)$ and $(0,1,0,0,1,1,0)$. They cor-

respond to the minimal cost bases $\{t_1, t_5, t_6\}$ et $\{t_2, t_5, t_6\}$ for KB.

4.3 τ-transformation .

From the practical side, in order to compute purifying sets of tests by taking advantage of algorithms for assumption-based reasoning, it can prove convenient to normalize the KB under consideration through the following τ-transformation:

Definition 8 (τ-transformation)
Let $KB = \langle W, \Delta \rangle$ an assumption-based theory where $\Delta = \{\phi_1, \ldots, \phi_n\}$. The τ-transformation of KB is the assumption-based theory $\tau(KB) = \langle \tau(W), \tau(\Delta) \rangle$ with $\tau(W) = W \cup \{ok(i) \Rightarrow \phi_i, bad(i) \Rightarrow \neg\phi_i, forget(i) \Leftrightarrow (\neg ok(i) \wedge \neg bad(i)) \mid i \in 1 \,..\, n\}$ where $ok(1), \ldots, ok(n)$, $bad(1), \ldots, bad(n)$, $forget(1), \ldots, forget(n)$ are new propositional symbols, not appearing in KB, and $\tau(\Delta) = \{ok(1), \ldots, ok(n)\}$.

Slightly abusing notations, we note $\tau(\phi_i) = ok(i)$ for every $\phi_i \in \Delta$, and $\tau(S) = \{\tau(\phi) \mid \phi \in S\}$ for every $S \subseteq \Delta$.

Example 1 (continued) $\tau(KB) = \langle \tau(W), \tau(\Delta) \rangle$ with

$\tau(W) =$ $(ok(1) \Rightarrow (a \wedge b)) \wedge (bad(1) \Rightarrow \neg(a \wedge b))$
$\wedge\ (forget(1) \Leftrightarrow (\neg ok(1) \wedge \neg bad(1)))$
$\wedge\ (ok(2) \Rightarrow \neg a) \wedge (bad(2) \Rightarrow a)$
$\wedge\ (forget(2) \Leftrightarrow (\neg ok(2) \wedge \neg bad(2)))$
$\wedge\ (ok(3) \Rightarrow \neg b \wedge c) \wedge (bad(3) \Rightarrow \neg(\neg b \wedge c))$
$\wedge\ (forget(3) \Leftrightarrow (\neg ok(3) \wedge \neg bad(3)))$
$\wedge\ (ok(4) \Rightarrow d) \wedge (bad(4) \Rightarrow \neg d)$
$\wedge\ (forget(4) \Leftrightarrow (\neg ok(4) \wedge \neg bad(4)))$ and

$\tau(\Delta) =$ $\{ok(1), ok(2), ok(3), ok(4)\}$.

The τ-transformation is a linear-time transformation that can be considered as a generalization of the notion of σ-transformation given in [8], better suited to the default revision situation. τ-transforming a KB offers many advantages. First, the set of assumptions of a τ-transformed KB is (by construction) a set of variables. This enables representing nogoods and maximal scenarios as terms. Second, it is just a naming trick, so it does not deeply change the nogoods, maximal scenarios and extensions. To be more precise, we need the notion of V-equivalence of two assumption-based theories.

Definition 9 (V-equivalence of two KBs)
*Let $KB_1 = \langle W_1, \Delta_1 \rangle$ and $KB_2 = \langle W_2, \Delta_2 \rangle$ be two assumption-based theories, and let $V \subseteq PS$. KB_1 and KB_2 are V-*equivalent*, noted $KB_1 \equiv_V KB_2$, iff there exists a bijective mapping M from $Ext(KB_1)$ to $Ext(KB_2)$ such that for every $E \in Ext(KB_1)$, $E \cap PROP_V = M(E) \cap PROP_V$.*

Proposition 11
Let $KB = \langle W, \Delta \rangle$ with $\Delta = \{\phi_1, \ldots, \phi_n\}$. Let $V = PS \setminus \cup_{i=1}^{n} \{ok(i), bad(i), forget(i)\}$. We have $KB \equiv_V \tau(KB)$. To be more precise, we have:

- *S is a nogood of KB iff $\tau(S)$ is a nogood of $\tau(KB)$.*
- *S is a maximal scenario of KB iff $\tau(S)$ is a maximal scenario of $\tau(KB)$.*
- *Let ψ be any formula not containing any variable from $\{ok(i), bad(i), forget(i)\}_{i \in 1..n}$ and let E_S be the extension of KB generated by the maximal scenario S. $\psi \in E_S$ iff $\psi \in E_{\tau(S)}$, where $E_{\tau(S)}$ is the extension of $\tau(KB)$ generated by $\tau(S)$.*

Additionally, while $\tau(KB)$ can have exponentially many nogoods and exponentially many maximal scenarios, there are situations in which computing $Nogood(\tau(KB))$ or $MaxScen(\tau(KB))$ is easy. Indeed, as the next proposition shows it, computing $MaxScen(\tau(KB))$ is tractable when $\tau(W)$ is a DNF formula. Moreover, when $\tau(W)$ is given by its prime implicates, computing $Nogood(\tau(KB))$ is tractable.

Proposition 12
Let $KB = \langle W, \Delta \rangle$ be an assumption-based theory.

- *If $\tau(W) = \psi_1 \vee \ldots \vee \psi_m$ is a DNF formula, then $MaxScen(\tau(KB)) = \{\{ok(i) \in \tau(\Delta) \mid \psi_j \wedge ok(i)$ satisfiable$\} \mid j \in 1..m\}_{\min \supset}$.*
- *If $\tau(W) = \psi_1 \wedge \ldots \wedge \psi_p$ is given by the conjunction of its prime implicates, then $Nogood(\tau(KB)) = \{S \subseteq \tau(\Delta) \mid \neg S$ (disjunctively interpreted) is a prime implicate of $\tau(W)\}$.*

Finally, contrariwise to σ-transformed KB, the default revision process for τ-transformed KB can be achieved through a simple expansion of $\tau(W)$.

Proposition 13
Let $KB = \langle W, \Delta \rangle$ with $\Delta = \{\phi_1, \ldots, \phi_n\}$. Let $V = PS \setminus \cup_{i=1}^{n} \{ok(i), bad(i), forget(i)\}$. For any i in $1...n$:

- *$KB * ok(i) \equiv_V \langle \tau(W) \cup \{ok(i)\}, \tau(\Delta) \rangle$.*
- *$KB * bad(i) \equiv_V \langle \tau(W) \cup \{bad(i)\}, \tau(\Delta) \rangle$.*
- *$KB * forget(i) \equiv_V \langle \tau(W) \cup \{forget(i)\}, \tau(\Delta) \rangle$.*

Accordingly, computing $KB * R$ simply consists in expanding W with R. This proves particularly valuable in the situation where W is a DNF formula (resp. the conjunction of its prime implicates). Indeed, in this situation, a DNF (resp. the set of prime implicates) of $W \cup R$ can be easily computed (see, e.g., [10]), hence it is also the case for $MaxScen(KB * R)$ (resp. $Nogood(KB * R)$). As a consequence, determining whether revising KB with R N-purifies KB can be decided tractably in these two restricted cases.

5 Designing default verification tests

In the formal framework developed in this paper, only tests enabling direct default validation, cancellation or invalidation are considered. At a first glance, this restriction may seem unrealistic since, in the general case, there might be actions of verification that do not check the default ϕ_i itself but a relevant formula (maybe weaker than ϕ_i, maybe stronger, maybe neither of both). For instance, Charlie (he's an engineer) wrote a program C that checks whether an integer is divisible by 2, 3, 5 or 7. A positive answer on 348754697 results in a formula which implies (but is not equivalent to) "348754697 is not prime". Moreover, if Charlie wrote as well a program D that tests whether an integer is divisible by an integer strictly greater than 5, then running C and D not only enables us determining whether "348754697 is prime" but gives us as well extra-information which might be relevant for another default of Δ.

In the following, we show how default verification tests can be generated from more general tests.

We propose the following propositional formulation of a test. More complex formulations can be found in the literature [13] [20] [22] [11].

Definition 10 (test)
A test t *is a (finite) set of propositional formulas* $\{\psi_1, \ldots, \psi_m\}$ *where* $m \geq 2$. *Each* ψ_i *represents a possible outcome of the test. To each test* t *is attached a positive integer* k, *the cost of performing* t.

A *test environment* is an ordered pair $\langle W, T \rangle$ where W is a finite set of propositional formulas (facts) and $T = \{t_1, \ldots, t_p\}$ is a finite set of available tests. We make the following assumptions:

- Tests always give a (unique) outcome[6].
- Tests are *reliable*: if the outcome of $t \in T$ is ψ_i, then ψ_i is true in I [7].
- Tests are *non-intrusive*: performing t does not change the actual world I. Subsequently, the order with respect to which tests are performed does not matter: the outcome R of a set $X \subseteq T$ of tests is simply the set of the outcomes of each individual test t of X. $State(X)$ is the set of all possible resulting outcomes of a set X of tests.

[6]This assumption is technical and does not mean that tests do not have preconditions or never fail: simply, when the test is not executable or when it fails, its outcome will be the tautology $\top$.

[7]We can assume without loss of generality that for all $i \in 1 .. m$, $W \cup \{\psi_i\}$ is satisfiable – otherwise, ψ_i will never be reached as an outcome and is thus irrelevant.

Note that we do not require tests to have exclusive outcomes. This enables taking into account tests that may succeed or fail in a graduate way. For instance, $t = \{a, a \vee b, \top\}$ used for determining the truth value of a can either succeed (the outcome is a), fail completely (the outcome is $\top$: no information) or only partially (the outcome $a \vee b$ is relevant to a). The verification actions defined in Section 3 are special cases of tests. Namely, $\{ok(i), bad(i)\}$, $\{ok(i), forget(i)\}$ and $\{ok(i), bad(i), forget(i)\}$ respectively correspond to the tests $\{\phi_i, \neg\phi_i\}$, $\{\phi_i, \top\}$ and $\{\phi_i, \neg\phi_i, \top\}$.

Sets of tests can be characterized according to their ability to help in determining the truth value of a given formula ϕ_i (for our purposes, ϕ_i is a default of Δ):

Definition 11 ((in)validation, discrimination)
Let $\langle W, T \rangle$ *a test environment,* $X \subseteq T$ *a set of tests, and* ϕ_i *a formula from* $PROP_{PS}$:

- X potentially validates ϕ_i *iff at least one of its outcomes* $R \in State(X)$ *is s.t.* $W \cup R \models \phi_i$. X potentially invalidates ϕ_i *iff it potentially validates* $\neg\phi_i$.
- X discriminates ϕ_i *iff for each of its outcomes* $R \in State(X)$, *we have either* $W \cup R \models \phi_i$ *or* $W \cup R \models \neg\phi_i$.

We now briefly show how to generate default verification tests from a set $T = \{t_1, \ldots, t_p\}$ of more "general" tests: if T discriminates ϕ_i (which is the most favorable case), then ϕ_i is associated to the test $\{ok(i), bad(i)\}$; else, if T potentially validates and potentially invalidates ϕ_i, then the associated test is $\{ok(i), bad(i), forget(i)\}$; if T only potentially validates ϕ_i, then the test is $\{ok(i), forget(i)\}$; if T does not potentially validate ϕ_i (even if it potentially invalidates it), then no default verification test is associated to ϕ_i (ϕ_i is not testable)[8].

When tests of T have costs, a default verification test generated from T as described above has an associated cost too. For instance, the cost of a $\{ok(i), bad(i)\}$ default verification test is the cost k of a set $X \subseteq T$ of minimal cost that discriminates ϕ_i (note that k is obtained by aggregating the costs of the individual tests of X). Several aggregation policies and a trade-off between the costs and the discrimination power of tests can be considered.

Example 5 *Let* $\langle W, T \rangle$ *be a test environment s.t.* $W = \{a \vee b, b \Leftrightarrow c, a \wedge b \Rightarrow f, b \vee g, a \vee \neg f, a \vee \neg g\}$

[8]It would not cause any technical difficulty to consider an action $\{bad(i), forget(i)\}$ when T only potentially invalidates ϕ_i; but, as explained in Section 3, the assumption that a default is plausible makes them of little relevance.

and $T = \{t_a, t_b, t_c\}$ *where* $t_a = \{a, \neg a\}$, $t_b = \{b, \neg b\}$ *and* $t_c = \{c, \neg c\}$. *Let* $\phi_1 = f$, $\phi_2 = f \vee g$, $\phi_3 = f \wedge \neg g$.

- *T both potentially validates and potentially invalidates ϕ_1 (consider respectively $R = \{a, b, c\}$ and $R' = \{\neg a, b, c\}$), but does not discriminate ϕ_1 (consider $R = \{a, \neg b, \neg c\}$).*
- *T discriminates ϕ_2. Since $W \models (\phi_2 \Leftrightarrow a)$, $\{t_a\}$ discriminates ϕ_2 (hence T discriminates ϕ_2 as well).*
- *T does not potentially validate ϕ_3 (but it potentially invalidates it).*

Clearly enough, the complexity of designing default verification tests from more general tests merely depends on the complexity of determining whether T potentially validates ϕ_i (resp. discriminates it). We call the corresponding decision problems VALIDABILITY (resp. DISCRIMINABILITY)[9].

Proposition 14

- VALIDABILITY *is* Π_2^p*-complete.*
- DISCRIMINABILITY *is* coNP*-complete.*

It is particularly interesting to note that discriminability (which is a stronger, and perhaps more useful notion than validability and invalidability) is also computationally easier. Introducing a name for each test outcome (under the form of a new variable) and binarizing tests with more than two possible outcomes enables characterizing the sets of tests we are looking for using prime implicates. The validability (resp. discriminability) issue is strongly related to logic-based abduction (resp. definability), and it is possible to take advantage of the complexity results from [5] (resp. [9]) to derive the results above.

6 Related work and conclusion

This paper is (to our knowledge) the first approach aiming at "debugging" default theories by performing appropriate default verification actions, but is is closely connected (and often complementary) to the following areas: tests in diagnosis, revision of default theories and inference from default theories.

The model-based diagnosis community has been interested for long in test selection strategies for isolating faulty components. Discriminating among several extensions and discriminating among plausible diagnoses are two almost identical tasks: the parallel between the complements of maximal scenarios (often called candidates) and preferred (consistency-based) diagnoses is well-known [18] [6]. A first class of approaches, along the works on GDE [4], consists in selecting the next best test to perform (*one-step lookahead* strategy) so as to maximize the expected gain of information. Much closer to our formalization of tests is the work of McIlraith and Reiter [12] [11] where tests are classified in function of their potentiality to confirm or discard one or several hypotheses; the search for "good" tests is characterized in abductive terms and consequently, test generation can be automated using results and algorithms for the computation of abductive explanations. This line of work (focusing on test generation) can be thought of as upstream whereas ours (rather concerned with what can be done with tests once they are fixed) is downstream.

Williams and Antoniou [23] propose a methodology for revising default theories. They consider four types of revision, namely (a) contraction (removal of a fact), (b) denial (ensuring that a given formula appears in no extension), (c) incorporation of information consistent with W and (d) incorporation of information inconsistent with W. Their revisions lead to change W while ours lead to change both Δ and W. Our validations, consisting of "moving" a formula from Δ to W, are a special case of (c) – no matter whether the default remains in Δ or not. Our invalidations are also a special case of (c) (note that we never question the truth of formulas of W, neither the outcomes of tests, which guarantees that test outcomes are consistent with W). Lastly, our cancellations are a different type of contraction than (a) and they induce a smaller amount of change than fact contraction.

There is a large amount of work on inference from default theories [2], and, relatedly, on belief base revision [14] and on reasoning from inconsistent knowledge bases [3]. There is much to say, first, on the effects of tests outcomes on inference: to be short, default validations strengthen skeptical inference and weaken credulous inference, cancellations weaken credulous inference and have various kinds of effects on skeptical inference (depending on the "role" played by the cancelled default before its cancellation); things are more complex with invalidations. Now, there is also much to say and to do about the *cooperation* between both tasks (inference and discrimination). Inference enables pointing on relevant tests: to give an example, it can be shown that a test $\{ok(i), forget(i)\}$ is useful for discrimination iff the corresponding ϕ_i belongs to at least one extension but not all, i.e., if it is credulously inferred but not skeptically. Other questions of interest are: until when should we discriminate among extensions so as to make the inference informa-

[9] Obviously, the complexity of INVALIDABILITY and of VALIDABILITY coincide.

tive enough and/or sound enough? Which verification actions should we focus on when we are no more interested in the whole purification task but rather in the inference (or not) of a particular formula? Some of our results can be considered as a first step towards these issues.

Acknowledgements

The second author has been partly supported by a "Contrat d'objectifs de la Région Nord/Pas-de-Calais" and by the IUT de Lens.

References

[1] G. Antoniou. On the dynamics of default reasoning. *Proc. of ECSQARU'99*, 1-10.

[2] G. Brewka. Preferred subtheories: an extended logical framework for default reasoning. *Proc. of IJCAI'89*, 1043-1048.

[3] S. Benferhat, C.Cayrol, D. Dubois, J. Lang and H. Prade. Inconsistency management and prioritized syntax-based entailment. *Proc. of IJCAI'93*, 640-645.

[4] J. de Kleer and B. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32, 97-130, 1987.

[5] Th. Eiter and G. Gottlob. The complexity of logic-based abduction. *JACM*, 42(1), 3-42, 1995.

[6] J. de Kleer, A. Mackworth and R. Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56, 197-222, 1992.

[7] G. Gottlob. Complexity results for nonmonotonic logics. *Journal of Logic and Computation*, 2, 397-425, 1992.

[8] A. Kean. A formal characterization of a domain independent abductive reasoning system. Ph.D. thesis, University of British Columbia, 1992.

[9] J. Lang and P. Marquis. Complexity results for independence and definability in propositional logic. *Proc. of KR'98*, 356-367.

[10] P. Marquis. Consequence finding algorithms. In vol. 5, *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, Kluwer Academic, 1999 (to appear).

[11] S. McIlraith. Generating tests using abduction. *Proc. of KR'94*, 449-460.

[12] S. McIlraith and R. Reiter. On tests for hypothetical reasoning. *Readings in model-based diagnosis*, Morgan Kaufman, 89-96, 1992.

[13] R.C. Moore. A formal theory of knowledge and action. *Readings in planning*, Morgan Kaufmann, 480-519, 1990.

[14] B. Nebel. Belief revision and default reasoning: syntax-based approaches. *Proc. of KR'91*, 417-428.

[15] Ch. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[16] D. Poole. A logical framework for default reasoning. *Artificial Intelligence* 36, 27-47, 1987.

[17] D. Poole. Normality and faults in logic-based diagnosis. *Readings in model-based diagnosis*, Morgan Kaufman, 71-77, 1992.

[18] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1), 57-96, 1987.

[19] R. Rymon. Search through systematic set enumeration. *Proc. of KR'92*, 539-550.

[20] R.B. Scherl and H.J. Levesque. The frame problem and knowledge-producing actions. *Proc. of AAAI'93*, 689-695.

[21] J. Stillman. The complexity of propositional default logics. *Proc. of AAAI'92*, 794-799.

[22] B. van Linder, W. van der Hoek and J.-J. Meyer. Tests as epistemic updates. *Proc. of ECAI'94*, 331-335.

[23] M.A. Williams and G. Antoniou. A strategy for revising default theory extensions. *Proc. of KR'98*, 24-33.

Ordering explanations and the structural rules for abduction

Ramón Pino-Pérez*
Centre de recherches en informatique de Lens
Faculté des Sciences
Université d'Artois
62307 Lens, France
pino@cril.univ-artois.fr

Carlos Uzcátegui
Departamento de Matemáticas
Facultad de Ciencias
Universidad de Los Andes
Mérida 5101, Venezuela
uzca@ciens.ula.ve

Abstract

We study the relationship between some structural rules for abductive reasoning and preference relations for selection preferred explanations. We prove that explanatory relations having good structural properties can always be defined by orders over formulas.

1 Introduction

Abduction is usually defined as the process of inferring the best explanation of an observation. In the logic-based approach to abduction, the background theory is given by a consistent set of formulas Σ. The notion of a *possible explanation* is defined by saying that a formula γ is an explanation of α if $\Sigma \cup \{\gamma\} \vdash \alpha$. An explanatory relation is a binary relation $\rhd$ where the intended meaning of $\alpha \rhd \gamma$ is "γ is a *preferred explanation* of α".

Structural properties for abduction has been studied by Flach [3], Cialdea-Mayer and Pirri [10], Aliseda [1]. The search for these properties is motivated by questions of the following type: (i) Suppose that γ is a preferred explanation of $\alpha \wedge \beta$. Should γ be considered also a preferred explanation of α? (ii) If γ is a preferred explanation of α and also of β, is γ a preferred explanation of $\alpha \vee \beta$? (iii) If γ is a preferred explanation of α and γ' entails γ, should γ' be considered a preferred explanation of α? Answers to these questions will tell how much a change of an observation affects its preferred explanations and more important will contribute to truly make abduction a form of logical inference. We have presented in [13, 14] a fairly complete list of rationality postulates for abduction in the Kraus-Lehmann-Magidor tradition.

A second natural question is whether changes in the background theory should be allowed during the explanatory process. We will not consider this interesting aspect in this paper (some remarks about it can be found in the introduction of [14]).

In this paper we will consider a third aspect of abduction: preference criteria for selecting explanations. Most formalism have treated them as external devices which work on top of the logical part of abduction. However, the exact relationship between the preference criteria and the logical or structural properties of the explanatory mechanism has not been so far clearly delineated. The main goal of this paper is to clarify this problem.

Perhaps the most natural way of defining an explanatory relation $\rhd$ is through a preference relation $\prec$ over formulas. The relation $\prec$ will tell which formulas in $Expla(\alpha)$ (the set of possible explanations of α) are the preferred ones. Let us define $\alpha \rhd \gamma$ iff γ is a $\prec$-minimal element of $Expla(\alpha)$. We will show the exact correspondence between general structural properties for $\rhd$ and the properties of the preference relation $\prec$ (i.e, properties like being modular, filtered, smooth, etc). In particular, we will show that the preference criteria is, in fact, implicit in the structural rules satisfied by a given explanatory mechanism. Moreover, we will show that this is necessarily the case: the logical properties satisfied by an explanatory relation $\rhd$ already encode a selection mechanism. These are the main results of this paper.

This paper is organized as follows: In Section 2 we recall the main structural rules introduced in [14] and the basic hierarchy of explanatory relations. Section 3 deals with explanatory relations defined by "orders" (selection mechanisms) over formulas. We define the essential relation and prove some basic representation theorems. Section 4 is devoted to study the role of the particular rule *Right And*. In Section 5 we present

* Visiting the Departamento de Matemáticas, Facultad de Ciencias, Universidad de Los Andes, Mérida 5101, Venezuela (until August 2000)

some examples. In Section 6 we compare briefly our work with other related works. We conclude with some remarks and open problems in Section 7.

2 Structural rules

The *background theory* will be a consistent set of formulas in a classical propositional language and will be denoted by Σ. Also, we will write $\alpha \vdash_\Sigma \beta$ when $\Sigma \cup \{\alpha\} \vdash \beta$. We could have avoided the use of Σ and $\vdash_\Sigma$ and instead use a semantic entailment relations $\models$ satisfying the standard requirements (like compactness and the properties of $\vee$ and $\wedge$). This way the background theory would be taken for granted and the notion of explanation would be somewhat elliptical. But we have chosen to keep Σ for several reasons. First of all, because it is customary in most presentation of abduction to have a background theory. Secondly, because many examples are naturally presented with a background theory that constrains the notion of explanation. And third, because by keeping Σ we leave open the question regarding the properties of abduction when the background theory is also considered as a parameter.

We now introduce the notion of an explanation of a formula with respect to Σ.

Definition 2.1 *For every formula α, the collection of explanations of α w.r.t. Σ is denoted by $Expla(\alpha)$ and is defined as follows:*

$$Expla(\alpha) = \{\gamma : \gamma \not\vdash_\Sigma \bot \ \ \& \ \ \gamma \vdash_\Sigma \alpha\}$$

Notice that we have ruled out trivial explanations by asking that γ has to be consistent with Σ. We are interested in studying the relation "γ is a preferred explanation of α", which will be denoted by $\alpha \rhd \gamma$. In explanatory reasoning the input is an observation and the output is an explanation, that is the reason to write $\alpha \rhd \gamma$ with α as input and γ as output. Our next definition capture the ideas mentioned in the introduction.

Definition 2.2 *Let Σ be a background theory. An* **explanatory relation** *for Σ will be any binary relation $\rhd$ such that for every α and γ,*

$$\alpha \rhd \gamma \ \ \Rightarrow \ \ \gamma \not\vdash_\Sigma \bot \ \ \textit{and} \ \ \gamma \vdash_\Sigma \alpha$$

We read $\alpha \rhd \gamma$ as saying that γ is a preferred explanation (with respect to Σ) of α.

The following rules were introduced in [14] and are the structural rules mentioned in the introduction. These rules are desirable since in a sense they describe properties of well behaved explanatory relations:

LLE$_\Sigma$: $$\frac{\vdash_\Sigma \alpha \leftrightarrow \alpha' \ ; \ \alpha \rhd \gamma}{\alpha' \rhd \gamma}$$

RLE$_\Sigma$: $$\frac{\vdash_\Sigma \gamma \leftrightarrow \gamma' \ ; \ \alpha \rhd \gamma}{\alpha \rhd \gamma'}$$

E-CM: $$\frac{\alpha \rhd \gamma \ ; \ \gamma \vdash_\Sigma \beta}{(\alpha \wedge \beta) \rhd \gamma}$$

E-C-Cut: $$\frac{(\alpha \wedge \beta) \rhd \gamma \ ; \ \forall \delta \, [\alpha \rhd \delta \ \Rightarrow \ \delta \vdash_\Sigma \beta]}{\alpha \rhd \gamma}$$

RA: $$\frac{\alpha \rhd \gamma \ ; \ \gamma' \vdash_\Sigma \gamma \ ; \ \gamma' \not\vdash_\Sigma \bot}{\alpha \rhd \gamma'}$$

LOR: $$\frac{\alpha \rhd \gamma \ ; \ \beta \rhd \gamma}{(\alpha \vee \beta) \rhd \gamma}$$

E-DR: $$\frac{\alpha \rhd \gamma \ ; \ \beta \rhd \delta}{(\alpha \vee \beta) \rhd \gamma \text{ or } (\alpha \vee \beta) \rhd \delta}$$

E-R-Cut: $$\frac{(\alpha \wedge \beta) \rhd \gamma \ ; \ \exists \delta \, [\alpha \rhd \delta \ \& \ \delta \vdash_\Sigma \beta]}{\alpha \rhd \gamma}$$

E-RW : $$\frac{\alpha \rhd \gamma \ ; \ \alpha \rhd \delta}{\alpha \rhd (\gamma \vee \delta)}$$

E-Con$_\Sigma$: $\not\vdash_\Sigma \neg\alpha$ iff there is γ such that $\alpha \rhd \gamma$

E-Reflexivity: $$\frac{\alpha \rhd \gamma}{\gamma \rhd \gamma}$$

Notice that some of these postulates (like E-C-Cut) are not rules in the usual finitary sense, since they have quantifiers in the premises. However, for the sake of simplicity, we will keep the standard notation used for rules in propositional logic.

The justification and intuition behind these rules were given in [14]. Nevertheless, for the sake of completeness, we will make some brief comments about these rules. The rules (LLE$_\Sigma$) *Left Logical Equivalence* and (RLE$_\Sigma$) *Right Logical Equivalence* are very natural assumptions. They say that explanatory relations are independent of the syntax. The rule (E-CM), *Explanatory Cautious Monotony*, expresses a weak form of a monotonicity on the left.

The rules (E-C-Cut), *Explanatory Cautious Cut*, and (E-R-Cut), *Explanatory Rational Cut* are the explanatory cut rules. They play an important role in our setting. Actually, there is a duality between monotony

rules for consequence relations and cut rules. They say that a preferred explanation of the more complex observation ($\alpha \wedge \beta$) might also be, in some cases, a preferred explanation of the simpler or incomplete observation (α). In other words, Cut rules allow to keep a preferred explanation even when the set of observations is not longer complete. As we will see, these rules reflect a selection mechanism. In fact, a failure of full cut (*i.e.* $\alpha \wedge \beta \rhd \gamma$ but $\alpha \not\rhd \gamma$) says that there is some part (β) of an observation ($\alpha \wedge \beta$) which is so relevant for explaining the whole observation that it can not be ignored. This difference between the whole observation and a part of it will be reflected in the selection of preferred explanations.

The rule RA, *Right And*, gives some amount of monotony on the right. A similar postulate has been considered by Flach in [3]. RA says that any explanation more "complete" (logically stronger, more specific) than a preferred explanation of α is also a preferred explanation of α. Notice that RA and E-RW are the only rules that introduce new explanations.

Let us remark some controversial aspects of RA. A typical example where one might not wish to have RA is something of the following sort: if *"there is sugar in the coffee"* is a preferred explanation of the observation *"the coffee is good"* then RA would declare that *"there are sugar and pepper in the coffee"* should also be a preferred explanation of the observation. It is in cases like this that the theory Σ can play an important role by ruling out such undesirable explanations. However, it is clear that RA fails when every observation has an unique (up to logical equivalence) preferred explanation. There are natural examples of such explanatory relations. In Section 4 we will study more deeply the rule RA.

The rule (LOR) is *Left Or*. The intuition behind this rule is the following. Suppose that when we observe either α or β (no matter which one) we are willing to accept that γ is a very likely explanation for both of them. Now we are told that one of them is observed (but maybe it is not known which one) then LOR says that it is rational to conclude that γ is still a very likely explanation of that observation (*i.e.* a very likely explanation of $\alpha \vee \beta$). The rule E-DR *Explanatory Disjunctive Rationality* is stronger than LOR and has a similar interpretation.

Finally E-Con$_\Sigma$, *Explanatory Consistency Preservation*, is the postulate that says when a formula has a preferred explanation: just when the observation is consistent with Σ.

Definition 2.3 *Let Σ be a background theory and $\rhd$ be an explanatory relation. We say that $\rhd$ is* **E-preferential** *if it satisfies* E-CM, E-C-Cut, LLE$_\Sigma$ *and* RA. *$\rhd$ is* **E-disjunctive rational** *if it is E-preferential and in addition satisfies* E-DR. *$\rhd$ is* **E-rational** *if it is E-preferential and in addition satisfies* E-R-Cut.

The basic motivation is the following. To each explanatory relation $\rhd$ we associate a consequence relation $\mid\!\sim_{ab}$ as follows:

$$\alpha \mid\!\sim_{ab} \beta \text{ if } \Sigma \cup \{\gamma\} \vdash \beta \text{ for each } \gamma \text{ such that } \alpha \rhd \gamma \quad (1)$$

We read $\alpha \mid\!\sim_{ab} \beta$ as "normally, when α is observed then β should also be present".

We proved in [14] that if $\rhd$ is E-preferential then $\mid\!\sim_{ab}$ is preferential (in the KLM sense [6]); if $\rhd$ is E-disjunctive rational then $\mid\!\sim_{ab}$ is disjunctive rational (*i.e.* in addition of the preferential rules the following also holds: if $\alpha \vee \beta \mid\!\sim \rho$ then $\alpha \mid\!\sim \rho$ or $\beta \mid\!\sim \rho$) and finally if $\rhd$ is E-rational then $\mid\!\sim_{ab}$ is rational (*i.e.* a preferential relation which satisfies also *rational monotony* :if $\alpha \mid\!\sim \rho$ and $\alpha \not\mid\!\sim \rho$ then $\alpha \wedge \beta \mid\!\sim \rho$). These notions form a hierarchy:

E-preferential $\subset$ E-disjunctive rational $\subset$ E-rational.

To each consequence relation $\mid\!\sim$ we associate an explanatory relation $\tilde{\rhd}$ by putting

$$\alpha \tilde{\rhd} \gamma \quad \text{iff } \gamma \not\vdash_\Sigma \bot \text{ and } C(\alpha) \subseteq Cn(\Sigma \cup \{\gamma\}) \quad (2)$$

For an adequate consequence relation $\mid\!\sim$ [1] we have shown [14] that if $\mid\!\sim$ is preferential then $\tilde{\rhd}$ is E-preferential; if $\mid\!\sim$ is disjunctive rational then $\tilde{\rhd}$ is E-disjunctive rational and if $\mid\!\sim$ is rational then $\tilde{\rhd}$ is E-rational.

These results show a formal duality between explanatory relations and consequence relations. We will give more details later in the paper.

3 Ordering explanations.

As we have said in the introduction the most distinct feature of abduction is the emphasis it makes on preferred explanations rather than plain explanations. In this section we will focus on preference criteria for defining explanatory relations. We will show

[1] $\mid\!\sim$ is said to be *adequate with respect to* Σ if for every formula α the following holds:

$$C(\alpha) = \bigcap \{Cn(\Sigma \cup \{\gamma\}) : C(\alpha) \subseteq Cn(\Sigma \cup \{\gamma\})\}$$

that these preference criteria are implicitly built in the structural properties of explanatory relations introduced in Section 2. These results are quite natural on the light of the well known facts about nonmonotonic reasoning. In fact, it is well known that inference processes based on orders over formulas are one the "faces" of non monotonic reasoning [11]. For instance possibilistic orders [2] and expectations orders [5] characterize inference rational relations. Preferential orders [4] characterize preferential relations. We will comment about their connection with our results.

We will start by making precise some basic notions. If $\prec$ is an irreflexive binary relation over a set S and $A \subseteq S$, then $a \in A$ is a $\prec$-*minimal* element of A if there is no $b \in A$ with $b \prec a$. The minimal elements of a set A will be denoted by $min(A, \prec)$ and when there is no confusion about which preference relation $\prec$ is used we will just write $min(A)$.

We formally define the notion of a preference relation.

Definition 3.1 *A* **preference relation** $\prec$ *will be any binary irreflexive relation $\prec$ over $\mathcal{L}$ which is invariant under logical equivalence w.r.t. Σ,* i.e. *if $\alpha \prec \beta$ and $\vdash_\Sigma \alpha \leftrightarrow \alpha'$ and $\vdash_\Sigma \beta \leftrightarrow \beta'$, then $\alpha' \prec \beta'$.*

To each preference relation $\prec$ we associate an explanatory relation $\rhd$ as follows:

Definition 3.2 *Let $\prec$ be an irreflexive relation on formulas. The explanatory relation $\rhd$ associated to $\prec$ is defined by*

$$\alpha \rhd \gamma \stackrel{def}{\Leftrightarrow} \gamma \in min(Expla(\alpha), \prec) \qquad (3)$$

i.e. *$\alpha \rhd \gamma$ iff $\not\vdash_\Sigma \neg\gamma$, $\gamma \vdash_\Sigma \alpha$ and $\delta \not\vdash_\Sigma \alpha$ for all δ such that $\delta \prec \gamma$.*

Definition 3.2 is the same one given in [8, 10] (but notice that they worked with reflexive relations). Notice also that $\prec$ is not supposed to be transitive, thus the notion of $\prec$-minimal element might not be intuitive. We will be interested mainly in the case where $\prec$ is at least smooth (see definition below). The two basic questions that we will address are then the following:

1. To determine the relationship between the structural properties of $\prec$ (like being a partial, filtered, modular order) and the structural rules satisfied by $\rhd$.
2. To determine under which conditions a given explanatory relation can be represented by a preference relation $\prec$ as in (3).

First, we point out a simple fact.

Proposition 3.3 *Let $\prec$ be a binary relation as in 3.2 and $\rhd$ be the corresponding explanatory relation. Then $\rhd$ satisfies* E-Reflexivity *and* E-CM.

Proof: To see E-Reflexivity just notice that if $\alpha \rhd \gamma$ then it is obvious that $\gamma \in min(Expla(\gamma)), \prec)$.To check that E-CM holds, suppose $\alpha \rhd \gamma$ and $\gamma \vdash_\Sigma \beta$, then $\gamma \in min(Expla(\alpha)) \cap Expla(\alpha \wedge \beta) \subseteq min(Expla(\alpha \wedge \beta))$. ∎

To obtain other postulates we will impose some constrains over $\prec$. The following notion of a smooth relation is inspired by the notion of smoothness used in the study of consequence relations ([6]).

Definition 3.4 *Let $\prec$ be a reflexive binary relation over a set S. We say that a subset $A \subseteq S$ is* **smooth** *if for every $a \in A$ either a is minimal in A or there is $b \in A$ with $b \prec a$ and b minimal in A. A preference relation $\prec$ as in 3.1 is called* **smooth**, *if for every formula α the set $Expla(\alpha)$ is smooth.*

To understand better the meaning of smoothness we remark the following: Let $A \subseteq B \subseteq S$, then it is clear that $min(B) \cap A \subseteq min(A)$. Suppose now that $min(B) \subseteq A$, hence $min(B) \subseteq min(A)$. It is reasonable then to expect that $min(A) = min(B)$. This is true when B is smooth since, in this case, $min(A) \subseteq min(B)$. Notice that when the language if finite every transitive relation $\prec$ is obviously smooth.

Theorem 3.5 *If $\prec$ is a smooth preference relation and $\rhd$ is defined as in (3.2), then $\rhd$ is an explanatory relation that satisfies* LLE_Σ, RLE_Σ, E-CM, E-C-Cut *and* $\mathsf{E\text{-}Con}_\Sigma$.

Proof: That LLE_Σ and RLE_Σ hold follows from the fact that $\prec$ is logically invariant. We already have shown in 3.3 that E-CM holds. To see that $\mathsf{E\text{-}Con}_\Sigma$ holds, suppose that α is consistent with Σ then $Expla(\alpha)$ is not empty. By smoothness there is γ such that $\alpha \rhd \gamma$. To see that E-C-Cut, suppose that the premises in the rule E-C-Cut hold. Hence $min(Expla(\alpha)) \subseteq Expla(\beta)$ and since $Expla(\alpha \wedge \beta) \subseteq Expla(\alpha)$, then $min(Expla(\alpha)) \subseteq min(Expla(\alpha \wedge \beta))$. Since $Expla(\alpha)$ is smooth we conclude $min(Expla(\alpha)) = min(Expla(\alpha \wedge \beta))$ and this finishes the proof. ∎

It seems natural to expect that under the conditions in the conclusion of 3.5 the relation $\rhd$ is represented by a smooth preference relation as in 3.2. However, in order to get such representation we will need more than just E-CM and E-C-Cut.

First, we introduce some necessary notions.

Definition 3.6 *Let $\rhd$ be an explanatory relation. We*

will say that a formula γ is admissible for $\rhd$ *if $\alpha \rhd \gamma$ for some formula α.*

The following definition, motivated by the results in [12], is the key point in order to get our basic representation theorem.

Definition 3.7 *Let $\rhd$ be an explanatory relation that satisfies* RLE_Σ. *The* essential preference relation *associated to $\rhd$ is denoted by $\prec_e$ and defined by:*

(a) For δ not admissible: $\gamma \prec_e \delta$ for every admissible γ.

(b) For γ and δ admissible: $\gamma \prec_e \delta$ if $Cn(\Sigma \cup \{\gamma\}) \cap \{\beta : \beta \rhd \delta\} = \emptyset$.

The only relevant formulas for the definition of $\prec_e$ are admissible formulas. Since $\rhd$ satisfies RLE_Σ then $\prec_e$ is invariant under logically equivalence and thus it is indeed a preference relation. Notice that admissible formulas are consistent with Σ. Admissible formulas play in our paper the same role as normal models in [6, 9]. A concept similar to that of an admissible formula was defined in [3].

Remark 3.8 *Suppose $\prec$ is a preference relation and $\rhd$ is the associated explanatory relation (3.2). It is easy to verify that if $\gamma \prec \delta$, then $\gamma \prec_e \delta$. In other words, $\prec_e$ is larger than $\prec$.*

The proof that $\prec_e$ represents $\rhd$ will work when the language is finite and more generally for explanatory relations which are logically finite either on the right or on the left (see definition below). First, we introduce an auxiliary notion.

Definition 3.9 *A set of formulas A is said to have an upper bound (in A w.r.t Σ) if there are finitely many formulas $\alpha_1, \ldots, \alpha_n \in A$ such that for all $\alpha \in A$, $\alpha \vdash_\Sigma (\alpha_1 \vee \cdots \vee \alpha_n)$ (i.e, $\alpha_1 \vee \cdots \vee \alpha_n$ is an upper bound of A in the lattice of formulas modulo Σ).*

Definition 3.10 *An explanatory relation $\rhd$ is said to be* **logically finite on the right** *(*RLF*) if for every formula α the set $\{\gamma : \alpha \rhd \gamma\}$ has an upper bound.*

Definition 3.11 *An explanatory relation $\rhd$ is said to be* **logically finite on the left** *(*LLF*) if for every admissible formula γ the set $\{\alpha : \alpha \rhd \gamma\}$ has an upper bound.*

Definition 3.12 *An explanatory relation $\rhd$ is said to be* **logically finite** *if it satisfies* RLF *or* LLF.

Notice that if the language is finite then every explanatory relation is logically finite. We will give two examples of logically finite relations:

Example 3.13 *Let $\vdash\!\!\sim$ be an adequate consequence relation and $\tilde{\rhd}$ be the explanatory relation associated to $\vdash\!\!\sim$, defined by (2). If $\tilde{\rhd}$ is logically finite on the right, then there is a map F from formulas into formulas such that $C(\alpha) = Cn(\Sigma \cup \{F(\alpha)\})$. In fact, for every α let $F(\alpha)$ be $\gamma_1 \vee \cdots \vee \gamma_n$ the upper bound for $\{\gamma : \alpha \rhd \gamma\}$ given by 3.10 (if α is inconsistent with Σ, then we let $F(\alpha)$ be $\bot$). Conversely, it is clear that if such function F exists then $\tilde{\rhd}$ satisfies* RLF.

Example 3.14 *We will present an example of a* LLF *explanatory relation $\tilde{\rhd}$. We define first an adequate rational relation $\vdash\!\!\sim$ as follows: Consider an infinite language $\mathcal{L} = \{p_1, p_2, \ldots\}$. Let $\mathcal{L}_n = \{p_1, p_2, \ldots, p_n\}$ and fix m models $M_1, \ldots, M_m$ for the language $\mathcal{L}_n$. Let $L'_1 \cdots, L'_k$ be a partition of $\{M_1, \ldots, M_m\}$ in k levels and let $L'_i = \{M_i^1, \ldots, M_i^{n_i}\}$ for $i = 1, \ldots, k$. Now consider the ranked model in the language $\mathcal{L}$ given by k levels $L_1, \ldots, L_k$, where $M \in L_i$ iff the restriction of M to $\mathcal{L}_n$ is in L'_i. Let γ_i^j be formulas in the language $\mathcal{L}_n$ such that $\mathrm{mod}(\gamma_i^j) = M_i^j$. For $r = 1, \ldots, k$ we let β_r be the following formula:*

$$\beta_r = \bigvee_{i=r}^{k} \left(\bigvee_{j=1}^{n_i} \gamma_i^j \right)$$

Let $\Sigma = \{\beta_1\}$. It is not hard to see that the rational relation $\vdash\!\!\sim$ generated by this model is adequate (with respect to Σ). Moreover, this ranked model is standard, i.e. *for every formula α, $\mathrm{mod}(C(\alpha)) = \mathrm{mod}(\alpha) \cap L_i$, where i is the first integer j such that $\mathrm{mod}(\alpha) \cap L_j \neq \emptyset$. It is easy to check that a formula γ is admissible iff $C(\gamma) = Cn(\Sigma \cup \{\gamma\})$. Let γ be an admissible formula, then there is r such that $\mathrm{mod}(\Sigma \cup \{\gamma\}) \subseteq L_r$. We claim that β_r is an upper bound for $\{\alpha : \alpha \tilde{\rhd} \gamma\}$. In fact, it is easy to see that $\mathrm{mod}(\beta_r) = \bigcup_{i=r}^{k} L_i$ and $\mathrm{mod}(C(\beta_r)) = L_r$. Hence $\beta_r \tilde{\rhd} \gamma$. Now, if $\alpha \tilde{\rhd} \gamma$, then $\mathrm{mod}(\Sigma \cup \{\alpha\}) \subseteq \bigcup_{i=r}^{k} L_i$. Thus $\alpha \vdash_\Sigma \beta_r$.*

The next theorem gives a characterization of those logically finite explanatory relations representable by preference relations.

Theorem 3.15 *Let $\rhd$ be a logically finite explanatory relation. The following are equivalent:*

(i) The relation $\rhd$ satisfies E-CM, LLE_Σ, RLE_Σ, E-C-Cut, $\mathsf{E\text{-}Con}_\Sigma$ *and* LOR.

(ii) There is a smooth preference relation $\prec$ such that

$$min(Expla(\alpha)) \cap min(Expla(\beta)) \subseteq min(Expla(\alpha \vee \beta)) \quad (4)$$

and for every formula α the following holds

$$\alpha \rhd \gamma \text{ iff } \gamma \in min(Expla(\alpha), \prec) \quad (5)$$

Proof: $(ii) \Rightarrow (i)$. By 3.5 we only need to show that $\rhd$ satisfies LOR. But this follows immediately from (4).

$(i) \Rightarrow (ii)$. We will show that $\prec_e$ works. We already have observed that since $\rhd$ satisfies RLE_Σ then $\prec_e$ is a preference relation. First, notice that (4) follows immediately from (5) and LOR.

We will show that (5) holds. Let us suppose that $\alpha \rhd \gamma$ and let $\delta \in Expla(\alpha)$, then $\alpha \in Cn(\Sigma \cup \{\delta\}) \cap \{\beta : \beta \rhd \gamma\}$. Therefore $\delta \not\prec_e \gamma$ and $\gamma \in min(Expla(\alpha))$. This shows that the *only if* in (5) holds.

Fix a formula α' consistent with Σ and let δ' be any formula in $Expla(\alpha')$. We will show that if $\alpha' \not\rhd \delta'$, then there is γ such that $\alpha' \rhd \gamma$ and $\gamma \prec_e \delta'$. In particular, this will prove that $\prec_e$ is smooth and also that the other direction in (5) holds. Suppose $\alpha' \not\rhd \delta'$. If δ' is not admissible, then there is nothing to show because of the definition of $\prec_e$ and $\mathsf{E\text{-}Con}_\Sigma$. Hence we will assume that δ' is admissible. By $\mathsf{E\text{-}Con}_\Sigma$ there is γ such that $\alpha' \rhd \gamma$, so let

$$C_{\alpha'} = \bigcap \{Cn(\Sigma \cup \{\gamma\}) : \alpha' \rhd \gamma\}$$

and

$$S = C_{\alpha'} \cup \{\neg\beta : \beta \rhd \delta'\}.$$

We claim that S is consistent. In fact, suppose towards a contradiction, that S is inconsistent. By compactness there are β_i's for $i = 1, \ldots, n$ such that $\beta_i \rhd \delta'$ and $(\beta_1 \vee \cdots \vee \beta_n) \in C'_\alpha$. Let $\beta = \beta_1 \vee \cdots \vee \beta_n$. By LOR we know that $\beta \rhd \delta'$. By E-CM we have that $(\alpha' \wedge \beta) \rhd \delta'$. Since $\beta \in C_{\alpha'}$, then by E-C-Cut we conclude $\alpha' \rhd \delta'$, which is a contradiction. Therefore S is consistent.

Since $\rhd$ is logically finite there are two cases to be considered:

(a) $\rhd$ satisfies RLF, *i.e.* for every formula α the set $A = \{\gamma : \alpha \rhd \gamma\}$ has an upper bound. Let $\gamma_i \in A$, $i \leq n$ be an upper bound for A. It is easy to check that

$$\begin{aligned} C_{\alpha'} &= \bigcap \{Cn(\Sigma \cup \{\gamma\}) : \gamma \in A\} \\ &= \bigcap \{Cn(\Sigma \cup \{\gamma_i\}) : i \leq n\} \\ &= Cn(\Sigma \cup \{(\gamma_1 \vee \cdots \vee \gamma_n)\}. \end{aligned}$$

Let N be a model of S, then there is i such that $N \models \Sigma \cup \{\gamma_i\}$. As N is also a model of $\{\neg\beta : \beta \rhd \delta'\}$, then it is clear that $\gamma_i \prec_e \delta'$.

(b) $\rhd$ satisfies LLF, *i.e.* for every admissible formula γ the set $\{\beta : \beta \rhd \gamma\}$ has a upper bound. Since δ' is admissible, let $\beta_1, \ldots, \beta_n$ be such that $\beta_i \rhd \delta'$ and $\beta \vdash_\Sigma \beta_1 \vee \cdots \vee \beta_n$ for every β such that $\beta \rhd \delta'$. Let $\beta' = \beta_1 \vee \cdots \vee \beta_n$, then by LOR $\neg\beta' \in S$. Since S is consistent then $\beta' \notin C_{\alpha'}$, hence there is γ such that $\alpha' \rhd \gamma$ and $\gamma \not\vdash_\Sigma \beta'$. Therefore $\gamma \not\vdash_\Sigma \beta$, for all β such that $\beta \rhd \delta'$, *i.e.* $\gamma \prec_e \delta'$. ∎

We will continue now analyzing the properties that $\prec_e$ has when $\rhd$ satisfies extra axioms. We postpone to Section 4 the analysis of the effect that RA has on $\prec_e$.

When $\rhd$ satisfies E-DR, then $\prec_e$ can be described in a different way (a similar idea was used in [9, 12]). Recall that from Section 2 we know that E-DR implies LOR. We introduce the following definition

Definition 3.16 *Let $\rhd$ be an explanatory relation that satisfies RLE_Σ. Define a binary relation $\prec_u$ by:*

(a) For δ not admissible: $\gamma \prec_u \delta$ for every admissible γ.

(b) For γ and δ admissible:

$$\gamma \prec_u \delta \stackrel{def}{\Leftrightarrow} \forall\alpha\forall\beta\,[\alpha \rhd \gamma \;\&\; \beta \rhd \delta \;\Rightarrow\; (\alpha \vee \beta) \rhd \gamma \;\&\; (\alpha \vee \beta) \not\rhd \delta]$$

Proposition 3.17 *Let $\rhd$ be an explanatory relation that satisfies LLE_Σ, RLE_Σ, E-CM, E-C-Cut, and E-DR. Then $\prec_e = \prec_u$. Moreover, $\prec_u$ (and therefore $\prec_e$) is transitive.*

Proof: $(\prec_e \subseteq \prec_u)$: This follows quite straightforward from the hypotheses.

$(\prec_u \subseteq \prec_e)$: Let γ, δ be admissible formulas with $\gamma \prec_u \delta$. Suppose, towards a contradiction, that there is β such that $\beta \rhd \delta$ and $\gamma \vdash_\Sigma \beta$. Let α be any formula such that $\alpha \rhd \gamma$. Since $\gamma \vdash_\Sigma \beta$, then by E-CM we have $(\alpha \wedge \beta) \rhd \gamma$. Since $\vdash ((\alpha \wedge \beta) \vee \beta) \leftrightarrow \beta$ and $\gamma \prec_u \delta$, then (by LLE_Σ) we conclude that $\beta \not\rhd \delta$, which is a contradiction.

To see that $\prec_u$ is transitive, let γ_i be formulas such that $\gamma_1 \prec_u \gamma_2$ and $\gamma_2 \prec_u \gamma_3$. Without lost of generality we can assume that each γ_i is admissible. Let α_i be formulas such that $\alpha_i \rhd \gamma_i$. By E-DR it suffices to show that $(\alpha_1 \vee \alpha_3) \not\rhd \gamma_3$. Suppose, towards a contradiction, that $(\alpha_1 \vee \alpha_3) \rhd \gamma_3$. Since $\gamma_2 \prec_u \gamma_3$, then by definition of $\prec_u$ we have $(\alpha_1 \vee \alpha_2 \vee \alpha_3) \rhd \gamma_2$ and $(\alpha_1 \vee \alpha_2 \vee \alpha_3) \not\rhd \gamma_3$. Since $\gamma_1 \prec_u \gamma_2$, then analogously we have $(\alpha_1 \vee \alpha_2 \vee \alpha_3) \rhd \gamma_1$ and $(\alpha_1 \vee \alpha_2 \vee \alpha_3) \not\rhd \gamma_2$, which is a contradiction. ∎

In [4] it was used a notion of filtered relation. We can adapt this notion to our context as follows:

Definition 3.18 *A preference relation $\prec$ is said to be* filtered *if for every α and every $\gamma, \gamma' \in Expla(\alpha)$ such that $\gamma \notin min(Expla(\alpha))$ and $\gamma' \notin min(Expla(\alpha))$, there is $\delta \in min(Expla(\alpha))$, such that $\delta \prec \gamma$ and $\delta \prec \gamma'$.*

Using an argument similar to that in the proof of 3.15 the following theorem can be proved:

Theorem 3.19 *If $\rhd$ is a logically finite explanatory relation that satisfies* RLE_Σ, E-CM, E-C-Cut, E-Con_Σ, E-RW *and* E-DR *then $\prec_u$ (therefore $\prec_e$) is filtered.*

4 The role of RA

In general, the information contained in an explanatory relation $\rhd$ could be lost when we move to its associate consequence relation $\mid\!\sim_{ab}$ (defined in (1)). However, as it was shown in [14], this is not the case for *causal* explanatory relations which are those satisfying the following condition:

$$\alpha \rhd \gamma \text{ iff } C_{ab}(\alpha) \subseteq Cn(\Sigma \cup \gamma) \qquad (6)$$

Where $C_{ab}(\alpha) = \{\beta : \alpha \mid\!\sim_{ab} \beta\}$. In other words, $\rhd$ is causal if $\rhd = \tilde{\rhd}$ where $\tilde{\rhd}$ is the explanatory relation associated to $\mid\!\sim_{ab}$ which justifies our claim that for causal explanatory relations $\mid\!\sim_{ab}$ and $\rhd$ contains the same information. For a finite language, we have shown that an explanatory relation is causal iff it satisfies RA and E-RW. It is then clear that RA must have a very distinct effect on $\prec_e$. Causal explanations relations are nonmonotonic reasoning in reverse [14]

The following proposition says that if RA holds, then $\prec_e$ satisfies almost all the properties of a *preferential pre-ordering* as defined by Freund in [4]. In Section 6 we will compare in more detail the properties of preferential orders and the essential relation $\prec_e$.

Proposition 4.1 *Let $\rhd$ be an explanatory relation that satisfies* RLE_Σ. *Then the following holds:*

(i) Let γ, γ' and δ be admissible formulas such that $\gamma \vdash \gamma'$. If $\gamma \prec_e \delta$, then $\gamma' \prec_e \delta$.

(ii) Let γ and δ be admissible formulas. If $(\delta \vee \gamma) \prec_e \gamma$, then $\delta \prec_e \gamma$.

(iii) Suppose that $\rhd$ also satisfies RA. *Let γ, γ' and δ be formulas such that $\gamma \not\vdash_\Sigma \bot$ and $\gamma \vdash \gamma'$. If $\delta \prec_e \gamma$, then $\delta \prec_e \gamma'$.*

Proof: To see (ii), suppose $\delta \not\prec_e \gamma$ and let β be such that $\delta \vdash_\Sigma \beta$ and $\beta \rhd \gamma$. Then clearly $\gamma \vee \delta \vdash_\Sigma \beta$ and thus $(\gamma \vee \delta) \not\prec_e \gamma$. The proof of (i) is similar. For (iii), suppose that $\delta \prec_e \gamma$. Thus by definition δ is admissible. If γ' is not admissible then by definition $\delta \prec_e \gamma'$. Now suppose that γ' is admissible and also, towards a contradiction, that $\delta \not\prec_e \gamma'$. Let β be such that $\beta \rhd \gamma'$ and $\delta \vdash_\Sigma \beta$. Since $\gamma \vdash_\Sigma \gamma'$ and $\gamma \not\vdash_\Sigma \bot$, we have by RA that $\beta \rhd \gamma$ and therefore $\delta \not\prec_e \gamma$. ∎

Remark: *A way of understanding part (iii) of the previous proposition is as follows: Assume that $\rhd$ satisfies* RLE_Σ *and* RA. *Let γ_1 and γ_2 be two admissible formulas such that $\gamma_1 \vee \gamma_2$ is also admissible. Let $\gamma' = \gamma_1 \vee \gamma_2$. It is easy to check using 4.1 that $\gamma_i \not\prec_e \gamma'$ and $\gamma' \not\prec_e \gamma_i$,* i.e. *for each i, γ_i and γ' are $\prec_e$-incomparable. But in fact, 4.1 (i) (resp. (iii)) says more, namely that every formula above (resp. below) γ_i is also above (resp. below) $\gamma_1 \vee \gamma_2$. So in some sense $\gamma_1 \vee \gamma_2$ contains the information "coded" by γ_1 and γ_2. Since explanatory relations are defined using $\prec$-minimal explanations it is clear the relevance of (iii).*

Property (iii) corresponds to RA and we will denote this property by C-U (Continuing Up). To define it formally, we say that a binary relation $<$ over formulas satisfies C-U if the following holds:

C-U $\gamma \not\vdash_\Sigma \bot$ & $\gamma \vdash \gamma'$ & $\delta < \gamma \Rightarrow \delta < \gamma'$

Proposition 4.2 *If $\prec$ is a preference relation satisfying* C-U *then the explanatory relation associated to $\prec$ (defined in 3.2) satisfies* RA.

Proof: Suppose that $\alpha \rhd \gamma$, $\gamma' \vdash_\Sigma \gamma$ and $\gamma' \not\vdash_\Sigma \bot$. We want to show that $\alpha \rhd \gamma'$, *i.e.* $\gamma' \in min(Expla(\alpha), \prec)$. Since $\gamma' \not\vdash_\Sigma \bot$ then it is clear that $\gamma' \in Expla(\alpha)$. For reductio, assume there is $\delta \in Expla(\alpha)$ such that $\delta \prec \gamma'$. By C-U and since $\gamma' \vdash_\Sigma \gamma$ we have $\delta \prec \gamma$ contradicting the minimality of γ in $Expla(\alpha)$. ∎

In the result that follows, it is interesting to notice that the hypothesis of logically finiteness is not needed. We will use this result in the sequel.

Proposition 4.3 *Let $\rhd$ be an E-rational explanatory relation satisfying* E-Con_Σ. *Then the following holds: for all admissible formulas γ and δ,*

$$\gamma \prec_u \delta \Leftrightarrow \exists\alpha\, \exists\beta\, [\, \alpha \rhd \gamma \;\&\; \beta \rhd \delta \;\&\; (\alpha \vee \beta) \rhd \gamma \;\&\; (\alpha \vee \beta) \not\rhd \delta\,] \qquad (7)$$

Moreover, $\prec_u$ (and therefore $\prec_e$) is smooth and represents $\rhd$.

Proof: The $\Rightarrow$ direction comes directly from the definition of $\prec_u$. For the other direction, let α and β be as in the right hand side of (7) and α' and β' be formulas such that $\alpha' \rhd \gamma$ and $\beta' \rhd \delta$. We need to show that $(\alpha' \vee \beta') \rhd \gamma$ and $(\alpha' \vee \beta') \not\rhd \delta$. Since $\rhd$ is E-rational, $\rhd$ satisfies E-DR (see [14]), hence it suffices to show that $(\alpha' \vee \beta') \not\rhd \delta$. Suppose, towards a contradiction, that $(\alpha' \vee \beta') \rhd \delta$. By E-CM we have $(\alpha' \vee \beta') \wedge (\alpha \vee \beta) \rhd \delta$. And by hypothesis $(\alpha \vee \beta) \rhd \gamma$

and clearly $\gamma \vdash_\Sigma (\alpha' \vee \beta')$, hence by E-R-Cut $(\alpha \vee \beta) \rhd \delta$, which contradicts the choice of α and β.

To see that $\prec_u$ is smooth, we first recall that $\rhd$ satisfies E-DR and therefore, by 3.17, $\prec_u = \prec_e$. As in the proof of 3.15 we have that if $\alpha \rhd \gamma$ then $\gamma \in min(Expla(\alpha), \prec_e)$. For the other direction, let $\delta \in Expla(\alpha)$ such that $\alpha \not\rhd \delta$. We will find γ such that $\alpha \rhd \gamma$ and $\gamma \prec_u \delta$. This will show that $\prec_u$ is smooth and also that it represents $\rhd$. We can assume without lost of generality that δ is admissible and thus let β be such that $\beta \rhd \delta$. Hence by E-CM $(\alpha \wedge \beta) \rhd \delta$. By E-Con$_\Sigma$ there is γ such that $\alpha \rhd \gamma$. Since α is logically equivalent to $(\alpha \wedge \beta) \vee \alpha$, then $((\alpha \wedge \beta) \vee \alpha) \not\rhd \delta$ and $((\alpha \wedge \beta) \vee \alpha) \rhd \gamma$. From (7) we conclude that $\gamma \prec_u \delta$. This finishes the proof. ∎

We will show next that when $\rhd$ satisfies E-R-Cut then $\prec_u$ is *modular*. We recall the definition of a modular relation (see [7]):

Definition 4.4 *A relation $\prec$ on a set E is said to be* modular *iff there exists a linear order $<$ on some set Ω and a function $r : E \to \Omega$ such that $a \prec b$ iff $r(a) < r(b)$. If $\prec$ is transitive, modularity is equivalent to the following property: for all a, b and c in E if a and b are $\prec$-incomparable and $a \prec c$ then $b \prec c$.*

Theorem 4.5 *Let $\rhd$ be an explanatory relation, the following are equivalent:*

(i) The relation $\rhd$ is E-rational and satisfies E-Con$_\Sigma$.

(ii) There is a smooth and modular preference relation $\prec$ satisfying C-U *such that for every α we have*

$$\alpha \rhd \gamma \quad \textit{iff} \quad \gamma \in min(Expla(\alpha), \prec)$$

Proof: (i ⇒ ii) From 3.15 we know that $\prec_e$ represent $\rhd$. From 4.1 we know that C-U holds. Thus, it remains to see that $\prec_e$ (alias $\prec_u$) is modular. Let γ, δ and ρ be formulas such that $\gamma \not\prec_u \delta$, $\delta \not\prec_u \gamma$ and $\gamma \prec_u \rho$. We want to show that $\delta \prec_u \rho$. Without lost of generality we can assume that γ, δ and ρ are admissible. Let α, β, ω formulas such that $\alpha \rhd \gamma$, $\beta \rhd \delta$ and $\omega \rhd \rho$. Since γ and δ are $\prec_u$-incomparable then from 4.3 it follows that $(\alpha \vee \beta) \rhd \gamma$ and $(\alpha \vee \beta) \rhd \delta$. Again by 4.3 it suffices to show that $(\alpha \vee \beta \vee \omega) \rhd \delta$ and $(\alpha \vee \beta \vee \omega) \not\rhd \rho$. By E-DR, which is true because $\rhd$ is E-rational, it is enough to show that $(\alpha \vee \beta \vee \omega) \not\rhd \rho$. Since $\gamma \prec_u \rho$, then by definition of $\prec_u$ we have $(\alpha \vee \beta \vee \omega) \rhd \gamma$ and $(\alpha \vee \beta \vee \omega) \not\rhd \rho$.

(ii ⇒ i) From 3.5 we know that $\rhd$ satisfies LLE$_\Sigma$, RLE$_\Sigma$, E-CM, E-C-Cut and E-Con$_\Sigma$. From 4.2 we obtain RA. It remains to be shown that E-R-Cut holds. Let α, β, γ and δ formulas such that $(\alpha \wedge \beta) \rhd \gamma$, $\alpha \rhd \delta$ and $\delta \vdash_\Sigma \beta$. We need to show that $\alpha \rhd \gamma$. Suppose, towards a contradiction, that $\alpha \not\rhd \gamma$. Since $\gamma \vdash_\Sigma \alpha$, then by smoothness and the definition of $\rhd$, there is δ' such that $\alpha \rhd \delta'$ and $\delta' \prec \gamma$. Since $\alpha \rhd \delta$ then $\delta \not\prec \delta'$ and $\delta' \not\prec \delta$. By E-CM $(\alpha \wedge \beta) \rhd \delta$ and by modularity, $\delta \prec \gamma$, which contradicts the hypothesis that $(\alpha \wedge \beta) \rhd \gamma$. ∎

5 Examples

Preferential models are perhaps the easiest way of defining structures for modelling various knowledge representation problems. We can also use them here to construct explanatory relations illustrating the properties of $\prec_e$.

We will work with a finite language. A *preferential model* [2] consists of a collection S of valuations and a partial order $\prec$ on S. In our case S will be the collection mod(Σ) of models of Σ. Given a formula α we define its minimal models as usual:

$$\min(\alpha) = \{N : N \models \Sigma \cup \{\alpha\} \& M \not\models \alpha \text{ for all } M \prec N\}$$

The relation $\prec$ over valuations is meant to capture the preferences of the agent and thus min(α) contains the most preferred or normal worlds where the observation α holds. Therefore we can use min(α) to capture also our preference over explanations. There are several ways of doing so. We will present three of them.

In order to get a more clear picture of the examples that follow it is also convenient to have in mind the consequence relation associated to the preferential model which is defined by

$$\alpha \mathrel{|\!\sim} \beta \text{ iff } \min(\alpha) \subseteq \text{mod}(\beta) \qquad (8)$$

The following definition is also useful

$$C(\alpha) = \{\beta : \alpha \mathrel{|\!\sim} \beta\}$$

$C(\alpha)$ contains the nonmonotonic consequences of α.

(1) Causal explanatory relations: Define an explanatory relation $\rhd_c$ as follows:

$$\alpha \rhd_c \gamma \overset{def}{\Leftrightarrow} mod(\Sigma \cup \{\gamma\}) \subseteq min(\alpha) \qquad (9)$$

for any pair of consistent (with Σ) formulas α and γ. In other words, an explanation of α is a preferred one if all its models are normal for α. Notice that $\alpha \rhd_c \gamma$ iff $C(\alpha) \subseteq Cn(\Sigma \cup \gamma)$ and $\mathrel{|\!\sim}_{ab} = \mathrel{|\!\sim}$. Hence we have

[2] For the more general definition of a preferential model see [6]

that $\rhd_c$ is equal to $\tilde{\rhd}$ for the consequence relation $\sim$ given in (8) and therefore $\rhd_c$ is a causal relation (as defined in (6)).

It is not difficult to show that this method always yields E-preferential explanatory relations that moreover satisfies LOR and E-RW. Since the hypothesis of theorem 3.15 holds then $\rhd_c$ is represented by its associated essential relation which will be denoted by $\prec_e^c$. We will make some remarks about this particular $\prec_e^c$.

Admissible formulas are given by the antichains of the preferential model (*i.e.* sets of mutually $\prec$-incompatible valuations). In other word, γ is $\rhd_c$-admissible iff $mod(\Sigma \cup \{\gamma\})$ is an antichain. And conversely, given a collection $\mathcal{A} \subseteq \mathrm{mod}(\Sigma)$ of mutually incompatible valuations, then any formula γ such that $\mathcal{A} = \mathrm{mod}(\gamma)$ is $\rhd_c$-admissible. For $\rhd_c$-admissible formulas γ and δ the following holds:

$$\gamma \prec_e^c \delta \text{ iff } \exists N \models \delta \, \exists M \models \gamma \text{ such that } M \prec N$$

Notice that, in general, the relation $\prec_e^c$ is clearly not transitive. In fact it is quite easy to find an example such that $\gamma \prec_e^c \delta$ and conversely $\delta \prec_e^c \gamma$. However, the representation theorem 3.15 guarantees that $\prec_e^c$ is smooth. In the case that the preferential model is filtered (for the definition see [4]) then $\prec_e^c$ is transitive (this follows from 3.17 and the fact that in this case E-DR holds).

(2) Strong epistemic explanatory relations: Consider now the following explanatory relation

$$\alpha \rhd_{se} \gamma \overset{def}{\Leftrightarrow} \gamma \vdash_\Sigma \alpha \;\&\; \min(\gamma) \subseteq \min(\alpha) \qquad (10)$$

In words, an explanation of α is a preferred one if all its minimal models are also minimal for α Notice that $\alpha \rhd_{se} \gamma$ iff $C(\alpha) \subseteq C(\gamma)$ and $\gamma \vdash_\Sigma \alpha$. More details and motivations about this notion are given in §4 of [14]. For instance, $\rhd_{se}$ satisfies LLE, RLE, E-CM, E-RW, E-C-Cut but RA does not hold. Notice that $\rhd_{se}$ is full reflexive, so every formula consistent with Σ is $\rhd_{se}$-admissible.

The relation $\prec_e^s$ is characterized in a way quite similar to that of $\prec_e^c$.

$$\gamma \prec_e^s \delta \text{ iff } \exists N \in \min(\delta) \, \exists M \models \gamma \text{ such that } M \prec N$$

So the crucial difference is the notion of an admissible formula. In general LOR might not hold for $\rhd_{se}$ (so theorem 3.15 does not apply) however $\rhd_{se}$ is represented by its associated essential relation $\prec_e^s$ *i.e.* $\alpha \rhd_{se} \gamma$ iff γ is a $\prec_e^s$-minimal explanation of α.

(3) NMC explanatory relations: Consider the following:

$$\alpha \rhd_{nc} \gamma \overset{def}{\Leftrightarrow} \gamma \vdash_\Sigma \alpha \;\&\; \mathrm{mod}(\gamma) \cap \min(\alpha) \neq \emptyset \qquad (11)$$

for any pair of consistent (with Σ) formulas α and γ. In words, an explanation of α is a preferred one if at least one of its models is minimal for α. Notice that this is equivalent to saying that $\gamma \vdash_\Sigma \alpha$ and $\alpha \not\sim \neg\gamma$, so we called $\rhd_{nc}$ *nonmonotonically consistent* explanatory relation. It is not difficult to show that $\rhd_{nc}$ satisfies LLE, RLE,E-CM, E-RW, E-C-Cut but RA does not hold. Notice that $\rhd_{nc}$ is full reflexive, so every formula consistent with Σ is $\rhd_{nc}$-admissible.

The essential relation $\prec_e^{nc}$ is characterized as follows.

$$\gamma \prec_e^{nc} \delta \text{ iff } \forall N \in \min(\delta) \, \exists M \in \min(\gamma) \text{ with } M \prec N$$

Notice that $\prec_e^{nc}$ is transitive. Similar to what happens with the strong epistemic relation, $\rhd_{nc}$ might not satisfies LOR (and hence theorem 3.15 does not apply) however it is representable by $\prec_e^{nc}$. Moreover, $\prec_e^{nc}$ is a well known order among formulas as we will see in the following section.

6 Related works

We will comment briefly in this section about the connection of our results and the works of Freund [4], Gärdenfors and Makinson [5] and Dubois and Prade [2].

Freund characterize preferential consequence relations in terms of 'preferential orders'. He called *preferential order* any relation $<$ on formulae satisfying the following four properties:

P_0: $\alpha < \bot$
P_1: If $\alpha \vdash \beta$, then (a) $\alpha < \gamma \Rightarrow \beta < \gamma$
(b) $\delta < \beta \Rightarrow \delta < \alpha$
P_2: If $\alpha < \gamma$ and $\alpha < \delta$, then $\alpha < \gamma \vee \delta$
P_3: If $\alpha \vee \beta < \beta$, then $\alpha < \beta$

The connection of preferential orders with preferential consequence relations is as follows: Given a preferential consequence relation $\sim$ define $<_\sim$ by letting $\alpha <_\sim \beta$ if $\alpha \vee \beta \sim \neg\beta$. And conversely, given a preferential order $<$ on formulae define a consequence relation $\sim_<$ by letting $\alpha \sim_< \beta$ if $\alpha < \alpha \wedge \neg\beta$. Freund showed that $<_\sim$ is a preferential order and $\sim_<$ is a preferential consequence relation. The connection of Freund's order with our work is the following: for the NMC explanatory relation defined in Section 5 we have that $\prec_e^{nc}$ is exactly Freund's relation.

In general, the essential relation $\prec_e$ associated to a given explanatory relation $\rhd$ satisfies P_0 when the

formulas α and β are admissible and, except for $P_1(b)$, the others properties are also satisfied by $\prec_e$ (this follows from 4.1). However, it is important to remark that in general $\prec_e$ is not transitive but Freund's relation is. The conditions C-U and $P_1(b)$ seem to play dual roles, however a complete classification of preference relations is still to be done.

Finally, the expectation orders of Gärdenfors and Makinson are modular and can be defined in terms of Freund's relation as follows: $<$ is an expectation order iff the dual relation $<^*$ is a modular preferential order, where $\alpha <^* \beta$ if $\neg\beta < \neg\alpha$. A similar result holds for the possibilistic order of Dubois and Prade.

7 Final remarks

Selection mechanisms are a fundamental part of abduction. However, most formalism have treated them as external devices which work on top of the logical part of abduction. We have shown that preference criteria are built in the structural properties of explanatory relations. Moreover, our results show that the preference criteria has to be somewhat uniform in order that an explanatory relation satisfies structural rules.

There are some natural questions suggested by our results. First of all, our representation theorem 3.15 is not optimal, since we have presented examples of explanatory relations which are representable by its associated essential relation but they do not satisfy LOR. Secondly, all examples we have examined so far are based on preferential models. It would be interesting, for a future work, to study preference relations defined in terms of a *simplicity* criteria, for instance, syntactic simplicity. Finally, it would be also interesting to find representation theorems capturing exactly the type of explanatory relations defined by (10) and (11).

References

[1] Atocha Aliseda-Llera. *Seeking Explanations: Abduction in Logic, Philosophy of Science and Artificial Intelligence.* PhD thesis, Stanford University, Department of Computer Science, 1998.

[2] D. Dubois and H. Prade. Epistemic entrenchment and possibilistic logic. *Artificial Intelligence*, 50:223–239, 1991.

[3] P. A. Flach. Rationality postulates for induction. In Yoav Shoham, editor, *Proc. of the Sixth Conference of Theoretical Aspects of Rationality and Knowledge (TARK96)*, pages 267–281, The Netherlands, March 17-20, 1996.

[4] M. Freund. Injective models and disjunctive relations. *Journal of Logic and Computation*, 3:231-247, 1993.

[5] P. Gärdenfors and D. Makinson. Nonmonotonic inferences based on expectations. *Artificial Intelligence*, 65:197–245, 1994.

[6] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167-207, 1990.

[7] D. Lehmann and M. Magidor. What does a conditional knowledge base entail? *Artificial Intelligence*, 55:1–60, 1992.

[8] H.J. Levesque. A knowledge level account of abduction. In *Proceedings of the eleventh International Joint Conference on Artificial Intelligence*, pages 1061–1067, Detroit, 1989.

[9] J. Lobo and C. Uzcátegui. Abductive consequence relations. *Artificial Intelligence*, 89:149-171, 1997.

[10] M. Cialdea Mayer and F. Pirri. Abduction is not deduction-in-reverse. *Journal of the IGPL*, 4:1-14, 1996.

[11] D. Makinson. Five faces of minimality. *Studia Logica*, 52:339–379, 1993.

[12] R. Pino Pérez and C. Uzcátegui. On representation theorems for non-monotonic inference relations. *Journal of Symbolic Logic*. To appear.

[13] R. Pino Pérez and C. Uzcátegui. Abduction vs. deduction in nonmonotonic reasoning. In J. Delgrande and M. Truszczynski, editors, *Proceedings of the Seventh International Workshop on Nonmonotonic Reasoning, NR'98*, pages 42-54, Trento, Italy, May 30-June 1, 1998.

[14] R. Pino-Pérez and C. Uzcátegui. Jumping to explanations vs. jumping to conclusions. *Artificial Intelligence*, 111(2):131–169, 1999.

Valuation-ranked Preferential Model

Zhaohui Zhu *
Computer Science Dept.
Nanjing University
of Aero. & Astr.
Nanjing, China, 210016

Bin Li
Computer Science Dept.
Nanjing University
of Aero. & Astr.
Nanjing, China, 210016

Shifu Cheng
State key lab. of novel
software technology
in Nanjing University
Nanjing, China, 210093

Wujia Zhu †
State key lab. of novel
software technology
in Nanjing University
Nanjing, China, 210093

Abstract

H.Bezzazi, D.Makinson and R.Pino Pérez discuss a number of non-Horn rules that are stronger than or incomparable with *rational monotony*. They establish a representation theorem for P+RM+RC and P+RM+WD in terms of *quasi – linear* model. Their results leave open the question of representation theorems for the weaker postulate sets P+RC and P+WD. This paper aims to solve these questions in the framework of finite propositional logic.

1 INTRODUCTION

Nonmonotonic reasoning is one of important research fields in AI. Researches in this filed can be broadly classified into two categories: those that proposed systems in which nonmonotonic inferences are performed and those that presented general framework in which nonmonotonic reasoning systems could be compared and classified. In the former category, the best known are probably: negation as failure, circumscription, the modal system of McDermott and Doyle, default logic and autoepistemic logic. In the latter category, Gabbay was probably first to suggest focusing the study of nonmonotonic logics on their inference relations [Gabbay 1985]. Inspired by Gabbay's work, there has been interest in researching into nonmonotonic inference relation from various angles.

Among them, Kraus, Lehmann and Magidor present the postulates for *preferential* inference in [Kraus, Lehmann and Magidor 1990]. These postulates are Horn conditions, that is of the form: if such and such pairs are in the inference relation, so too is such another pair. Lehmann and Magidor also study a non-Horn condition called *rational monotony*. *Rational monotony* implies certain other non-Horn conditions of interest, such as *disjunctive rationality* and *negation rationality* [Lehmann and Magidor 1992]. Freund and Lehmann provide a semantic characterization of inference relations satisfying these two conditions [Freund and Lehmann 1996].

Makinson draw attention to one such rule, called *determinacy preservation*, showing that it lies between *monotony* and *rational monotony*, but without studying it semantically [Makinson 1993] . Bezzazi and Pino Pérez begin a semantic investigation of two other non-Horn rules, called *rational transitivity* and *rational contraposition* [Bezzazi and Pérez 1996]. Bezzazi, Makinson and Pino Pérez study these and related conditions more systematically, establishing interrelations and providing semantic characterizations in [Bezzazi, Makinson and Pérez 1997]. In particular, they establish the representation theorem for **P+RM+RC** and **P+RM+WD** (see next section for the definition of these systems), however, their result leave open the question of representation theorems for the weaker postulate sets **P+WD** and **P+RC**. In this paper, we will solve these questions in the framework of finite propositional logic.

The rest of the paper is organized as follows: In section 2, we recall some basic definitions and results related to this paper. In section 3, we introduce the definition of *valuation – ranked preferential* model. In section 4 and 5, we establish the representation theorems for **P+WD** and **P+RC**, respectively. In section 6, we deal with another open question posed in [Bezzazi, Makinson and Pérez 1997].

2 BACKGROUND

In this section, we will recall some basic definitions and results from [Kraus, Lehmann and Magidor 1990], [Lehmann and Magidor 1992] and [Bezzazi, Makinson and Pérez 1997], which will be used in this paper.

*State key lab. of novel software technology in Nanjing University, Nanjing, China, 210093. Email: b4894956@jlonline.com

†Department of computer science, Nanjing University of Aeronautics and Astronautics, Nanjing, China, 210016

2.1 SOME HORN AND NON-HORN RULES

We consider formulae of classical propositional calculus built over a set of atomic formulae denoted L plus two constants $\top$ and $\bot$ (the formulae *true* and *false* respectively). Let $Form(L)$ be the set of all well formed formulaes. If L is finite we will say that the propositional language is finite. Let U be the set of valuations, i.e. functions v: $L \cup \{\top, \bot\} \rightarrow \{0,1\}$ such that $v(\top) = 1$ and $v(\bot) = 0$. We use lower case letters of the Greek alphabet to denote formulae, and the letters v, v_1, $v_2 \cdots$ to denote valuations. As usually, $\vdash \alpha$ means that α is a tautology and $v \models \alpha$ means that v satisfies α where compound formulae are evaluated as usually.

We consider certain binary relations between formulae. These relations will be called inference relations or consequence relations. Gabbay uses the relation symbol $|\sim$ to denote nonmonotonic consequence to distinguish it from monotonic logical consequence. If α, β are formulas, then the sequence $\alpha \,|\sim\, \beta$ is called a conditional assertion. In [Gabbay 1985], a consequence relation is defined as any binary relation R between propositional formulas for which certain properties hold. If a pair $(\alpha, \beta) \in R$, then using this notion of consequence, one may sensibly conclude β given α, and write $\alpha \,|\sim\, \beta$. $\alpha \,|\not\sim\, \beta$ means $(\alpha, \beta) \notin R$. Certain especially interesting properties of sets of conditional assertions are described as follows, the intuition behind those rules may be found in [Freund and Lehmann 1996], [Makinson 1993] and [Bezzazi, Makinson and Pérez 1997].

LLE (Left Logical Equivalence)

$$\frac{\vdash \alpha \leftrightarrow \beta, \alpha \,|\sim\, \gamma}{\beta \,|\sim\, \gamma}$$

RW (Right Weakening)

$$\frac{\vdash \alpha \rightarrow \beta, \gamma \,|\sim\, \alpha}{\gamma \,|\sim\, \beta}$$

OR

$$\frac{\beta \,|\sim\, \gamma, \alpha \,|\sim\, \gamma}{\beta \vee \alpha \,|\sim\, \gamma}$$

AND

$$\frac{\alpha \,|\sim\, \gamma, \alpha \,|\sim\, \beta}{\alpha \,|\sim\, \gamma \wedge \beta}$$

CM (Cautious Monotony)

$$\frac{\alpha \,|\sim\, \beta, \alpha \,|\sim\, \gamma}{\beta \wedge \alpha \,|\sim\, \gamma}$$

Reflexivity

$$\alpha \,|\sim\, \alpha$$

NR (Negation Rationality)

$$\frac{\alpha \wedge \gamma \,|\not\sim\, \beta, \alpha \wedge \neg\gamma \,|\not\sim\, \beta}{\alpha \,|\not\sim\, \beta}$$

DR (Disjunctive Rationality)

$$\frac{\alpha \,|\not\sim\, \beta, \gamma \,|\not\sim\, \beta}{\alpha \vee \gamma \,|\not\sim\, \beta}$$

RM (Rational Monotony)

$$\frac{\alpha \wedge \gamma \,|\not\sim\, \beta, \alpha \,|\not\sim\, \neg\gamma}{\alpha \,|\not\sim\, \beta}$$

DP (Determinacy Preservation)

$$\frac{\alpha \wedge \gamma \,|\not\sim\, \neg\beta, \alpha \,|\sim\, \beta}{\alpha \wedge \gamma \,|\sim\, \beta}$$

RT (Rational Transitivity)

$$\frac{\alpha \,|\sim\, \beta, \beta \,|\sim\, \gamma, \alpha \,|\not\sim\, \neg\gamma}{\alpha \,|\sim\, \gamma}$$

RC (Rational Contraposition)

$$\frac{\alpha \,|\sim\, \beta, \neg\beta \,|\not\sim\, \alpha}{\neg\beta \,|\sim\, \neg\alpha}$$

WD (Weak Determinacy)

$$\frac{\alpha \,|\not\sim\, \beta, \top \,|\sim\, \neg\alpha}{\alpha \,|\sim\, \neg\beta}$$

M (Monotony)

$$\frac{\alpha \,|\sim\, \beta}{\alpha \wedge \gamma \,|\sim\, \beta}$$

A relation $|\sim$ is said to be *preferential* inference relation iff it contains all instances of **Reflexivity** axiom and is closed under the rules of **LLE**, **RW**, **AND**, **CM** and **OR**. All these rules will be abbreviated by **P**. A relation $|\sim$ is said to be *rational* iff it is *preferential* inference relation and satisfies **RM**.

Bezzazi, Makinson and Pino Pérez compare the strength of the rules **DP**, **RT**, **RC**, **WD** and **RM**, and prove the following proposition:

Proposition 2.1 [Bezzazi, Makinson and Pérez 1997] Given the *preferential* rules **P**, the rules **DP** and **RT** are equivalent, and are implied by **M**. They are also equivalent to the pair { **RM, RC** } and also to the pair { **RM, WD** }. Moreover, given **P**, **RC** implies both

WD and **NR**. However, given **P**, none of the following implications hold: **RM** to **WD**, **RC** to **DR**, **WD** to **NR**. □

Together with the result in [Makinson 1993] that **M** implies **DP** but not conversely, and that **RM** implies **DR** which implies **NR** but neither conversely, the above proposition give us the following diagram (see figure 1)[Bezzazi, Makinson and Pérez 1997], where one condition implies another, given a *preferential* inference relation, iff one can follow arrows from the former to the latter.

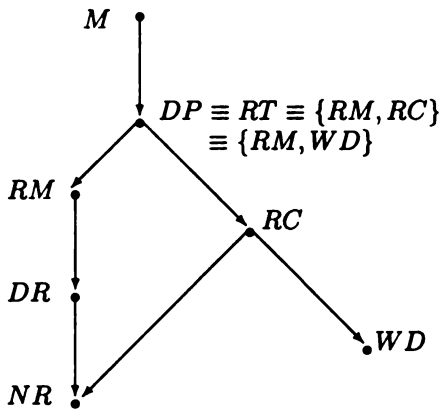


Figure 1: The Strength of Some Rules

2.2 SEMANTIC CHARACTERIZATIONS FOR SOME NONMONOTONIC CONSEQUENCE RELATIONS

Definition 2.1 [Kraus, Lehmann and Magidor 1990] A *preferential* model W is a triple $< S, l, \prec >$, where S is a set, the elements of which will be called states, $l : S \rightarrow U$ assigns a valuation to each state, where U is the set of all valuations, and $\prec$ is a strict partial order on S (i.e. transitive and irreflexive) satisfying the following smoothness condition: for each $\alpha \in Form(L)$, the set $\| \alpha \|_W = \{s : s \in W \text{ and } l(s) \models \alpha\}$ is smooth[1]. If there is no ambiguity, we shall write $\| \alpha \|$ instead of $\| \alpha \|_W$. □

Definition 2.2 [Lehmann and Magidor 1992] A *ranked* model is a *preferential* model $W = < S, l, \prec >$ such that there exists a totally strict order set $< \Phi, \rhd >$ and a function $f : S \rightarrow \Phi$ such that $s \prec t$ iff $f(s) \rhd f(t)$. □

Definition 2.3 Let $W = < S, l, \prec >$ be a *preferential* model, the inference relation generated by W will be denoted by $|\sim_W$ and is defined as follows:
$\alpha |\sim_W \beta$ iff for any s minimal in $\| \alpha \|$, $l(s) \models \beta$. □

Lehmann, Magidor and others have investigated the semantical characterization of *preferential* relation and rational relation in [Kraus, Lehmann and Magidor 1990] and [Lehmann and Magidor 1992]. In particular, they have established the representation theorems for them respectively.

Theorem 2.1[Kraus, Lehmann and Magidor 1990] $|\sim$ is a *preferential* inference relation iff there is a *preferential* model $W = < S, l, \prec >$ such that $|\sim = |\sim_W$. □

Theorem 2.2[Lehmann and Magidor 1992] $|\sim$ is a *rational* inference relation iff there is a *ranked* model $W = < S, l, \prec >$ such that $|\sim = |\sim_W$. □

Definition 2.4[Bezzazi, Makinson and Pérez 1997] A *preferential* model $W = < S, l, \prec >$ is said to be *quasi − linear* iff it is *ranked* and it has at most one state at any level above the lowest. In other words *quasi − linear* means *ranked* and whenever $r \prec s$, $r \prec t$ then either $s = t$ or $s \prec t$ or $t \prec s$. □

The following (representation) theorem is due to Bezzazi,Makinson and Pino Pérez :

Theorem 2.3[Bezzazi, Makinson and Pérez 1997] The following conditions are equivalent for any *preferential* inference relation $|\sim$:

1. $|\sim$ is generated by some *quasi − linear* model.
2. $|\sim$ is **determinacy preserving.**
3. $|\sim$ is **rational transitive.**
4. $|\sim$ satisfies both **RM** and **RC.**
5. $|\sim$ satisfies both **RM** and **WD.** □

The above theorem establish the representation theorem for **P+RM+RC** and **P+RM+WD**, however, leave open the question of representation theorems for the weaker postulate sets **P+RC** and **P+WD** (see [Bezzazi, Makinson and Pérez 1997]). This is one of open questions posed by Bezzazi, Makinson and Pino Pérez in [Bezzazi, Makinson and Pérez 1997]. In the following, we will solve the question in the framework of finite propositional logic.

3 VALUATION-RANKED PREFERENTIAL MODEL

Let $W = < S, l, \prec >$ be a *preferential model*. We adopt the following notations: the range of l will be denoted by $rang(l)$ (i.e. $rang(l) =_{def} \{v : \exists s(s \in S \text{ and } l(s) = v)\}$) . If $X \subseteq S$, then $\min(X)$ is the set of all minimal element of X with respect to $\prec$ (i.e. $\min(X) =_{def} \{t \in X : \neg\exists s(s \in X \text{ and } s \prec t)\}$), $l(X) =_{def} \{v : \exists s(s \in X \text{ and } l(s) = v)\}$. If $\Sigma \subseteq$

[1] Let W be a set ,$\prec$ be a strict partial order on W and $V \subseteq W$, we shall say that V is smooth iff for each $t \in V$, either t is itself minimal in V (i.e. there is no $w \in V$ such that $w \prec t$), or there exists $s \in V$ such that $s \prec t$ and s is minimal in V.

$rang(l)$, then $l^{-1}(\Sigma) =_{def} \{s \in S : \exists v(v \in \Sigma \text{ and } l(s) = v)\}$, we shall write $l^{-1}(v)$ instead of $l^{-1}(\{v\})$.

Definition 3.1 Let $W = < S, l, \prec >$ be a *preferential* model, the binary relation $\sqsubset$ is defined as follows: for any $X_1, X_2 \subseteq S$, $X_1 \sqsubset X_2$ iff $\forall s(s \in X_2 \Rightarrow \exists t(t \in X_1 \text{ and } t \prec s))$. □

Fact 3.1 Let $W = < S, l, \prec >$ be a *preferential* model, then $\sqsubset$ is transitive. □

Definition 3.2 A *preferential* model $W = < S, l, \prec >$ is said to be *valuation – ranked* iff $\sqsubset\Downarrow \{l^{-1}(v) : v \in rang(l) - l(\min(S))\}$ [2]is a linear order . In other words, for any $v_1, v_2 \in rang(l) - l(\min(S))$, if $v_1 \neq v_2$ then $l^{-1}(v_1) \sqsubset l^{-1}(v_2)$ or $l^{-1}(v_2) \sqsubset l^{-1}(v_1)$. □

4 REPRESENTATION THEOREM FOR P+WD

In this and later sections, our framework is finite propositional logic.

Definition 4.1 A relation $|\sim$ is said to be **P+WD** relation iff it is *preferential* and satisfies **WD**. □

Lemma 4.1 If Σ is a nonempty set of valuations, v is a valuation and $v \notin \Sigma$, then there exists a formula α such that $\Sigma \models \alpha$ and $v \models \neg\alpha$.

Proof We may suppose $\Sigma = \{v_1, v_2, \ldots v_n\}$ (Notice that we have assumed that our framework is finite propositional logic.). By $v \notin \Sigma$, we have $v \neq v_i$ $(i \leq n)$. Thus, there exist $\alpha_1, \alpha_2 \ldots \alpha_n$ such that $v_i \models \alpha_i$ and $v \models \neg\alpha_i$ for each $i \leq n$. Let $\alpha = \alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n$. Obviously $\Sigma \models \alpha$ and $v \models \neg\alpha$. □

Lemma 4.2 If $W = < S, l, \prec >$ is a *valuation – ranked preferential* model, then $|\sim_W$ is a **P+WD** inference relation.

Proof It is obvious that $|\sim_W$ is *preferential*, we want to show $|\sim_W$ satisfies **WD.** Suppose there exist α, β such that $\top |\sim_W \neg\alpha$, $\alpha \not|\sim_W \neg\beta$ and $\alpha \not|\sim_W \beta$. Hence, there are $s, s_1 \in \min(\| \alpha \|)$ such that $l(s) \models \beta$ and $l(s_1) \models \neg\beta$. Since $\top |\sim_W \neg\alpha$, we have $l(s) \notin l(\min(S))$ and $l(s_1) \notin l(\min(S))$. It is easy to show that $l(s) \neq l(s_1)$. Furthermore, by *valuation – ranked*, one of the following conditions holds:

(1) $l^{-1}(l(s_1)) \sqsubset l^{-1}(l(s))$.

(2) $l^{-1}(l(s)) \sqsubset l^{-1}(l(s_1))$.

However, they both contradict the fact that $s, s_1 \in \min(\| \alpha \|)$. Hence, there are no α, β such that $\top |\sim_W \neg\alpha$, $\alpha \not|\sim_W \beta$ and $\alpha \not|\sim_W \neg\beta$. Thus, $|\sim_W$ is a **P+WD** inference relation. □

[2] $\sqsubset\Downarrow \{l^{-1}(v) : v \in rang(l) - l(\min(S))\}$ is the restriction relation of $\sqsubset$ with respect to $\{l^{-1}(v) : v \in rang(l) - l(\min(S))\}$, i.e. $\sqsubset\Downarrow \{l^{-1}(v) : v \in rang(l) - l(\min(S))\} =_{def} \sqsubset \cap \{l^{-1}(v) : v \in rang(l) - l(\min(S))\}^2$.

Definition 4.2[Bezzazi, Makinson and Pérez 1997] A model $W = < S, l, \prec >$ is said to be *parsimonious* iff for every state $s \in S$ there is a formula α such that $s \in \min(\| \alpha \|)$. □

Lemma 4.3 If a *parsimonious preferential* model $W = < S, l, \prec >$ is not *valuation – ranked*, then $|\sim_W$ does not satisfy **WD** .

Proof Since $W = < S, l, \prec >$ is not *valuation – ranked*, there exist valuation $v_1, v_2 \in rang(l)$ satisfying the following conditions:

(1) $v_1 \neq v_2$.

(2) $v_i \notin l(\min(S))$ for $i = 1, 2$.

(3) $\exists s_1 \in l^{-1}(v_1) \forall t \in l^{-1}(v_2)(t \not\prec s_1)$.

(4) $\exists s_2 \in l^{-1}(v_2) \forall t \in l^{-1}(v_1)(t \not\prec s_2)$.

Let $\Omega = \min(S) \cup \{t : t \prec s_1\} \cup \{t : t \prec s_2\}$. By (2), (3), (4) and W being *parsimonious*, we have $v_i \notin l(\Omega)$ for $i = 1, 2$. By lemma 4.1, there is γ_i such that $v_i \models \gamma_i$ and $l(\Omega) \models \neg\gamma_i$ for $i = 1, 2$. Hence, $\top |\sim_W \neg(\gamma_1 \vee \gamma_2)$ and $s_1, s_2 \in \min(\| \gamma_1 \vee \gamma_2 \|)$. Since $v_1 \neq v_2$, there exists β such that $v_1 \models \beta$ and $v_2 \models \neg\beta$. Furthermore, we have $\gamma_1 \vee \gamma_2 \not|\sim_W \beta$ and $\gamma_1 \vee \gamma_2 \not|\sim_W \neg\beta$. Thus, $|\sim_W$ does not satisfy **WD.** □

The following theorem is due to Bezzazi, Makinson and Pino Pérez [Bezzazi, Makinson and Pérez 1997]:

Theorem 4.1[Bezzazi, Makinson and Pérez 1997] If $W = < S, l, \prec >$ is a *preferential* model then there exists a *preferential* model $W_1 = < S_1, l_1, \prec_1 >$ such that $S_1 \subseteq S$ and the following properties hold:

1. W_1 is *parsimonious*.

2. If W is *ranked* so is W_1.

3. Whenever $s, t \in S_1$ with neither $s \prec_1 t$ nor $t \prec_1 s$, if $l_1(s) = l_1(t)$ then $l(s) = l(t)$.

4. $|\sim_W = |\sim_{w_1}$. □

Theorem 4.2 (Representation theorem for P+WD) $|\sim$ is a **P+WD** inference relation if and only if there is a *valuation – ranked preferential* model $W = < S, l, \prec >$ such that $|\sim = |\sim_W$.

Proof ($\Rightarrow$) Since $|\sim$ is a **P+WD** inference relation, by theorem 2.1 and theorem 4.1, there exists a *parsimonious preferential* model $W = < S, l, \prec >$ such that $|\sim = |\sim_W$. So, $|\sim_W$ is a **P+WD** inference relation. Furthermore, by lemma 4.3, $W = < S, l, \prec >$ is *valuation – ranked.*
($\Leftarrow$) By lemma 4.2 . □

5 REPRESENTATION THEOREM FOR P+RC

Definition 5.1 A relation $|\sim$ is said to be **P+RC** inference relation iff it is *preferential* and satisfies

RC. □

Definition 5.2 A *preferential* model $W = < S, l, \prec >$ is said to be *PRC* model iff it satisfies the following conditions:

1.$W = < S, l, \prec >$ is *valuation − ranked.*

2.For each $v \in l(\min(S))$ and $s \in S$, if $l(s) \notin l(\min(S))$ and $l^{-1}(l(s))$ is not the minimum element of the linear order $\sqsubset\Downarrow \{l^{-1}(v) : v \in rang(l) - l(\min(S))\}$ [3] then there exists $t \in S$ such that $t \prec s$ and $l(t) = v$.

3.If $l^{-1}(v_0)$ is the minimum element of the linear order $\sqsubset\Downarrow \{l^{-1}(v) : v \in rang(l) - l(\min(S))\}$, then, for each $v \in l(\min(S))$ such that $\exists t(\ t \in l^{-1}(v_0)$ and $v \notin l(\{w : w \prec t\}))$, there exists $s \in l^{-1}(v_0)$ such that $l(\{t : t \prec s\}) = \{v\}$. □

Lemma 5.1 If $W = < S, l, \prec >$ is a *PRC* model, then $|\sim_W$ is a **P+RC** inference relation.

Proof It is obvious that $|\sim_W$ is *preferential*, we want to show $|\sim_W$ satisfies **RC.** Suppose there exist α, β such that $\alpha\ |\sim_W \beta$, $\neg\beta\ |\not\sim_W \alpha$ and $\neg\beta\ |\not\sim_W \neg\alpha$. Hence, there exist $s, x \in \min(\|\ \neg\beta\ \|)$ such that $l(x) \models \neg\alpha$ and $l(s) \models \alpha$. By $\alpha\ |\sim_W \beta$, we have $\top\ |\sim_W \alpha \rightarrow \beta$. So, $l(s) \notin l(\min(S))$. We consider three cases.

First, suppose $l(x) \notin l(\min(S))$. Since $l(s) \neq l(x)$ and $l(s) \notin l(\min(S))$, by *valuation−ranked*, we know that one of the following properties holds:

(1) $l^{-1}(l(x)) \sqsubset l^{-1}(l(s))$;

(2) $l^{-1}(l(s)) \sqsubset l^{-1}(l(x))$.

However, they both contradict $s, x \in \min(\|\ \neg\beta\ \|)$.

Second, suppose $l(x) \in l(\min(S))$ and $l^{-1}(l(s))$ is not the minimum element of the linear order $\sqsubset\Downarrow \{l^{-1}(v) : v \in rang(l) - l(\min(S))\}$. By the condition (2) in the definition of *PRC* model, there exists $t \in S$ such that $l(t) = l(x)$ and $t \prec s$, contradicting $s \in \min(\|\ \neg\beta\ \|)$.

Third, suppose $l(x) \in l(\min(S))$ and $l^{-1}(l(s))$ is the minimum element of the linear order $\sqsubset\Downarrow \{l^{-1}(v) : v \in rang(l) - l(\min(S))\}$. Since $s \in \min(\|\ \neg\beta\ \|)$ and $l(x) \models \neg\beta$, we have $l(x) \notin l(\{w : w \prec s\})$. By the condition (3) in the definition of *PRC* model, there exists $t \in l^{-1}(l(s))$ such that $l(\{w : w \prec t\}) = \{l(x)\}$. From $l(x) \models \neg\alpha$ and $l(s) \models \alpha$, we have $t \in \min(\|\ \alpha\ \|)$. This contradicts $\alpha\ |\sim_W \beta$.

Putting these three cases together, we see that there are no α, β such that $\alpha\ |\sim_W \beta$, $\neg\beta\ |\not\sim_W \alpha$ and $\neg\beta\ |\not\sim_W \neg\alpha$. Hence, $|\sim_W$ satisfies **RC** . □

Lemma 5.2 Let $W = < S, l, \prec >$ be a *parsimonious preferential* model that is *valuation − ranked.* Suppose that $s \in S$, $l(s) \in rang(l) - l(\min(S))$ and $l^{-1}(l(s))$ is not minimum element of the linear order $\sqsubset\Downarrow \{l^{-1}(v) : v \in rang(l) - l(\min(S))\}$. If there exists $v_1 \in l(\min(S))$ such that $v_1 \notin l(\{t : t \prec s\})$, then $|\sim_W$ does not satisfy **RC.**

Proof It is easy to show that $l(s) \neq v_1$. Let $\Sigma = \{v : v \in l(\min(S))$ and $v \neq v_1\} \cup \{l(t) : t \prec s\}$. Because of $v_1 \notin l(\{t : t \prec s\})$ and $l(s) \neq v_1$, we have $v_1 \notin \Sigma \cup \{l(s)\}$. Thus, by lemma 4.1, there exist α, γ_1 such that $\Sigma \models \alpha$, $l(s) \models \alpha$, $v_1 \models \neg\alpha$, $\Sigma \models \gamma_1$ and $v_1 \models \neg\gamma_1$. On the other hand, since W is *parsimonious preferential* model and $l(s) \in rang(l) - l(\min(S))$, we have $l(s) \notin \Sigma$. Hence, by lemma 4.1, there exists γ_2 such that $\Sigma \models \gamma_2$ and $l(s) \models \neg\gamma_2$. Let $\beta = \gamma_1 \wedge \gamma_2$. Then clearly we have $s \in \min(\|\ \neg\beta\ \|)$ and $l^{-1}(v_1) \cap \min(S) \subseteq \min(\|\ \neg\beta\ \|)$. Hence, $\neg\beta\ |\not\sim_W \neg\alpha$ and $\neg\beta\ |\not\sim_W \alpha$. In the following, we will show $\alpha\ |\sim_W \beta$. Suppose $t \in \min(\|\ \alpha\ \|)$. We consider two cases.

First, suppose $l(t) \in l(\min(S))$. Since $l(t) \models \alpha$ and $v_1 \models \neg\alpha$, we have $l(t) \neq v_1$ and $l(t) \in \Sigma$. Hence, $l(t) \models \beta$.

Second, suppose $l(t) \notin l(\min(S))$. We will verify that $l^{-1}(l(s)) \not\sqsubset l^{-1}(l(t))$ and $l(s) \neq l(t)$.
From $l(s) \models \alpha$ and $t \in \min(\|\ \alpha\ \|)$, we get $l^{-1}(l(s)) \not\sqsubset l^{-1}(l(t))$.
Assume $l(s) = l(t)$. Since $l^{-1}(l(s))$ is not the minimum element of the linear order $\sqsubset\Downarrow \{l^{-1}(v) : v \in rang(l) - l(\min(S))\}$, there exists $v_0 \in rang(l) - l(\min(S))$ such that $l^{-1}(v_0) \sqsubset l^{-1}(l(s))$. It is easy to show that $v_0 \models \alpha$ and $l^{-1}(v_0) \sqsubset l^{-1}(l(t))$, this contradicts $t \in \min(\|\ \alpha\ \|)$. Hence, $l(s) \neq l(t)$.
Furthermore, by *valuation − ranked*, we have $l^{-1}(l(t)) \sqsubset l^{-1}(l(s))$. By the construction of β we get $l(t) \models \beta$.

From the above two cases, we get $\alpha\ |\sim_W \beta$. Hence, there exist α, β such that $\alpha\ |\sim_W \beta$, $\neg\beta\ |\not\sim_W \neg\alpha$ and $\neg\beta\ |\not\sim_W \alpha$. Thus, $|\sim_W$ does not satisfy **RC.** □

Lemma 5.3 Let $W = < S, l, \prec >$ be a *parsimonious preferential* model that is *valuation − ranked.* Suppose that $v_0 \in rang(l) - l(\min(S))$ and $l^{-1}(v_0)$ is the minimum element of the linear order $\sqsubset\Downarrow \{l^{-1}(v) : v \in rang(l) - l(\min(S))\}$, and suppose that there exists $v_1 \in l(\min(S))$ satisfying the following conditions:

(1) $\exists x \in l^{-1}(v_0)(v_1 \notin l(\{w : w \prec x\}))$.

(2)$\neg\exists t \in l^{-1}(v_0)(l(\{s : s \prec t\}) = \{v_1\})$.

then $|\sim_W$ does not satisfy **RC.**

Proof It is easy to show that $v_0 \neq v_1$. Suppose $t \in l^{-1}(v_0)$ such that $v_1 \notin l(\{w : w \prec t\})$. Let $\Sigma = \{v : v \in l(\min(S))$ and $v \neq v_1\}$. Hence $v_0 \notin \Sigma$ and $v_1 \notin \Sigma \cup \{v_0\}$. Furthermore, by lemma 4.1, there exist α, γ_1 and γ_2 such that $\Sigma \models \alpha$, $v_0 \models \alpha$, $v_1 \models \neg\alpha$, $\Sigma \models \gamma_1$, $v_0 \models \neg\gamma_1$, $\Sigma \models \gamma_2$ and $v_1 \models \neg\gamma_2$. Let $\beta = \gamma_1 \wedge \gamma_2$. Since W is *parsimonious* and *valuation −ranked*, and $l^{-1}(l(t))$ is the minimum element of $\sqsubset\Downarrow \{l^{-1}(v) : v \in rang(l) - l(\min(S))\}$, it is easy to show that $l(\{w :$

[3] Let W be a set ,$\prec$ be a partial order on W and $V \subseteq W$, we shall say that $t \in V$ is a minimum of V iff for every $s \in V$, $s \neq t$, we have $t \prec s$. If t is a minimum of W , we also say that t is the minimum of $\prec$.

$w \prec t)\} \subseteq l(\min(S))$[4]. Furthermore, by $v_1 \notin l(\{w : w \prec t\})$, we have $t \in \min(\| \neg\beta \|)$. Thus $\neg\beta \not|\sim_W \neg\alpha$. From $l^{-1}(v_1) \cap \min(S) \neq \emptyset$ and $l^{-1}(v_1) \cap \min(S) \subseteq \min(\| \neg\beta \|)$, we have $\neg\beta \not|\sim_W \alpha$. In the following, we will show $\alpha |\sim_W \beta$. Suppose $w \in \min(\| \alpha \|)$.

We will verify $l(w) \in l(\min(S))$. Suppose $l(w) \notin l(\min(S))$. It is not hard to see that $l(w) \neq v_0$ [5]. Moreover, since $W = < S, l, \prec >$ is *valuation − ranked* and $l^{-1}(v_0)$ is the minimum element of the linear order $\sqsubset\Downarrow \{l^{-1}(v) : v \in rang(l) - l(\min(S))\}$, we have $l^{-1}(v_0) \sqsubset l^{-1}(l(w))$. Because of $v_0 \models \alpha$, this contradicts $w \in \min(\| \alpha \|)$. Hence, $l(w) \in l(\min(S))$. Since $l(w) \models \alpha$ and $v_1 \models \neg\alpha$, we have $l(w) \neq v_1$ and $l(w) \in \Sigma$. Hence, $l(w) \models \beta$. Furthermore, we get $\alpha |\sim_W \beta$.

Hence, there exist α, β such that $\alpha |\sim_W \beta$, $\neg\beta \not|\sim_W \neg\alpha$ and $\neg\beta \not|\sim_W \alpha$. Thus, $|\sim_W$ does not satisfy **RC**. □

Lemma 5.4 If a *parsimonious preferential* model $W = < S, l, \prec >$ is not *PRC* model then $|\sim_W$ does not satisfy **RC**.

Proof Since $W = < S, l, \prec >$ is not *PRC* model, W is not *valuation − ranked* or does not satisfy the condition (2) or (3) in the definition of *PRC* model. Suppose W is not *valuation − ranked*. By lemma 4.3, $|\sim_W$ does not satisfy **WD**. It is well known that **P+RC⇒WD** (see [Bezzazi, Makinson and Pérez 1997]), hence $|\sim_W$does not satisfy **RC**. Suppose W does not satisfy the condition (2) or (3) in the definition of *PRC* model. By Lemma 5.2 and 5.3, $|\sim_W$ does not satisfy **RC**. □

Theorem 5.1 (Representation theorem for P+RC) $|\sim$ is a **P+RC** inference relation if and only if there exists a *PRC* model $W = < S, l, \prec >$ such that $|\sim = |\sim_W$.

Proof ($\Rightarrow$) Since $|\sim$ is a **P+RC** inference relation, by theorem 2.1 and theorem 4.1, there exists a *parsimonious preferential* model $W = < S, l, \prec >$ such that $|\sim = |\sim_W$. So, $|\sim_W$ is a **P+RC** inference relation. Furthermore, by lemma 5.4, $W = < S, l, \prec >$ is a *PRC* model.
($\Leftarrow$)By lemma 5.1 . □

Definition 5.3 [Bezzazi, Makinson and Pérez 1997] A *preferential* model $W = < S, l, \prec >$ is said to be *injective* model iff l is *injective*. □

Theorem 5.2 [Bezzazi, Makinson and Pérez 1997] If $W = < S, l, \prec >$ is a *ranked* model then there exists a *parsimonious, injective ranked* model $W_1 = < S_1, l_1, \prec_1 >$ such that $|\sim_{W_1} = |\sim_W$ □

Corollary 5.1 [6] $|\sim$ is a **P+RT** inference relation if and only if there exists an *injective PRC* model $W = < S, l, \prec >$ such that $|\sim = |\sim_W$.
Proof ($\Rightarrow$) Suppose $|\sim$ is a **P+RT** inference relation, by theorem 2.3, definition 2.4 and theorem 5.2, there exists a *parsimonious, injective quasi − linear* model $W = < S, l, \prec >$ such that $|\sim = |\sim_W$. It is easily to show that $W = < S, l, \prec >$ is an *injective PRC* model.
($\Leftarrow$) Suppose $W = < S, l, \prec >$ is an *injective PRC* model and $|\sim = |\sim_W$. Thus, $l^{-1}(v)$ is a single set for each $v \in rang(l)$. So, by *valuation − ranked*, it has at most one state at any level above the lowest. By the condition (2) and (3) in the definition of *PRC* model and $W = < S, l, \prec >$ being *injective*, it is easily to show that $s \prec t$ for each $s \in min(S)$ and $t \in S - min(S)$. Constructing a totally strict order set $< \Phi, \rhd >$ and function $f : S \to \Phi$ as follows :

(1) $\Phi = (S - min(S)) \cup \{s_0\}$, where $s_0 \notin S$;

(2) $\rhd = \prec\Downarrow (S - min(S)) \cup \{< s_0, t >: t \in S - min(S)\}$;

(3) for each $s \in S$, $f(s) = \begin{cases} s & \text{if } s \in S - min(S) \\ s_0 & \text{if } s \in min(S) \end{cases}$

Obviously, $s \prec t$ iff $f(s) \rhd f(t)$ for each $s, t \in S$. Hence, $W = < S, l, \prec >$ is *quasi − linear* model. By theorem 2.3, $|\sim$ is a **P+RT** inference relation. □

6 A NOTE ON P+WD⇏NR

Bezzazi, Makinson and Pérez have showed that **P+WD⇏NR** in [Bezzazi, Makinson and Pérez 1997]. At the same time, they posed the following open question: Determine whether the non implication **P+WD⇏NR** can be witnessed by *injective preferential* models. In the finite language case, we will give negative answer by the following theorem.

Theorem 6.1 Let $W = < S, l, \prec >$ be an *injective preferential* model, if $|\sim_W$ is a **P+WD** inference relation then it satisfies **NR**.

Proof Suppose $|\sim_W$ is a **P+WD** inference relation and does not satisfy **NR**. Thus, there exist α, β and γ such that $\alpha |\sim_W \beta$, $\alpha \wedge \neg\gamma \not|\sim_W \beta$ and $\alpha \wedge \gamma \not|\sim_W \beta$. So, there exist $s \in \min(\| \alpha \wedge \neg\gamma \|)$ and $x \in \min(\| \alpha \wedge \gamma \|)$ such that $l(s) \models \neg\beta$ and $l(x) \models \neg\beta$. Obviously, $l(s) \neq l(x)$. Since $\alpha |\sim_W \beta$, we have $\top |\sim_W \alpha \to \beta$. Hence, $l(s), l(x) \notin l(\min(S))$. It is not hard to see that in the finite language, if a model is *injective* then the model is *parsimonious*. Since $|\sim_W$ is a **P+WD** inference relation, by lemma 4.3, $W = < S, l, \prec >$ is *valuation − ranked*. Furthermore, by *injective* and

[4]Otherwise, there exists $w \in S$ such that $w \prec t$ and $l(w) \in rang(l) - l(\min(S))$. Since $W = < S, l, \prec >$ is *parsimonious*, $l(w) \neq l(t)$. By *valuation − ranked*, it is easily to show $l^{-1}(l(w)) \sqsubset l^{-1}(l(t))$. This contradicts the fact that $l^{-1}(l(t))$ is the minimum element of $\sqsubset\Downarrow \{l^{-1}(v) : v \in rang(l) - l(\min(S))\}$.

[5]Otherwise, by $\neg\exists t \in l^{-1}(v_0)(l(\{s : s \prec t\}) = \{v_1\})$, we have $l(\{s : s \prec w\}) \cap (l(\min(S)) - \{v_1\}) \neq \emptyset$. Furthermore, there exist x such that $x \in \min(S)$, $x \prec w$ and $l(x) \models \alpha$, contradicting $w \in \min(\| \alpha \|)$.

[6]This corollary is due to the anonymous referee who pointed out this result in his comments.

valuation – ranked, we have $s \prec x$ or $x \prec s$. From $\alpha \mid\sim_W \beta$, $l(s) \models \alpha \wedge \neg\beta$ and $l(x) \models \alpha \wedge \neg\beta$, we know $s, x \in \| \alpha \|$ and $s, x \notin \min(\| \alpha \|)$. So, by the smoothness condition and ($s \prec x$ or $x \prec s$), there exists $y \in \min(\| \alpha \|)$ such that $y \prec s$ and $y \prec x$. Since $l(y) \models \alpha \wedge \gamma$ or $l(y) \models \alpha \wedge \neg\gamma$, it contradicts that $s \in \min(\| \alpha \wedge \neg\gamma \|)$ and $x \in \min(\| \alpha \wedge \gamma \|)$. Hence, there are no α, β and γ such that $\alpha \mid\sim_W \beta$, $\alpha \wedge \gamma \mid\not\sim_W \beta$ and $\alpha \wedge \neg\gamma \mid\not\sim_W \beta$. So, $\mid\sim_W$ satisfies **NR**. □

Remark: Schlechta proves that **P+ WD+ Injectivity** implies **RC** (even in the infinite cases), this result is stronger than the above theorem. On the other hand, he constructs an *injective*, non-smooth *preferential* model validating **Cumulativity** and **WD**, in which **NR** fails. Technical details may be found in [Schlechta 1999].

Acknowledgements

This authors would like to thank two anonymous referees for their comments. This research was partly funded by the NSF of China , NSF of Jiangsu Province and High Technology Research and Development program of china.

References

D.M.Gabbay (1985). Theoretical foundation for non-monotonic reasoning in expert systems. In: K.R.Apt. eds, *Proceeding NATO Advanced study Institute on logics and Models of Concurrent systems*, La Colle-syr-loup, France (Springer, Berlin,1985), 439-457.

S.Kraus, D.Lehmann and M.Magidor (1990). Non-monotonic reasoning, *preferential* models and cumulative logics. *Artif. Intell.* 44 (1-2), 1990, 167-207.

D.Lehmann and M.Magidor (1992). What does a conditional knowledge base entail ? *Artif. Intell.* 55 (1992), 1-60.

M.Freund and D.Lehmann (1996). On negation rationality. *Journal of Logic and Computation*, 6, 1-7, 1996.

D.Makinson (1993). General patterns in nonmonotonic reasoning, chapter 2 of *Handbook of logic in Artificial Intelligence and logic programming*, Volume III, Clarendon Press, Oxford, 1993.

H.Bezzazi and R.Pino Pérez (1996). Rational transitivity and its models. In *proceedings of the twenty-sixth International Symposium on Multiple-Valued logic*, 160-165, IEEE computer Society Press, 1996.

H.Bezzazi, D.Makinson and R.Pino Pérez (1997). Beyond rational monotonicity: Some strong non-Horn rules for nonmonotonic inference relations. *Journal of Logic and Computation*, 7 (1997), 605-631.

Schlechta (1999). A topological construction of non-smooth model of cumulativity. *Journal of Logic and Computation*, 9 (1999), 457-462.

Planning

Planning as Satisfiability with Expressive Action Languages: Concurrency, Constraints and Nondeterminism

Enrico Giunchiglia
DIST — Università di Genova
Viale Causa 13, 16145 Genova, Italy

Abstract

Planning as satisfiability [Kautz and Selman, 1992] is a very efficient technique for classical planning, i.e., for planning domains in which both the effects of actions and the initial state are completely specified. In this paper, we present a SAT-based procedure capable to deal with planning domains having incomplete information about the initial state, and whose underlying transition system is specified using the highly expressive action language $\mathcal{C}$ [Giunchiglia and Lifschitz, 1998]. Thus, the presented procedure allows for planning in domains involving (*i*) actions which can be executed concurrently; (*ii*) (ramification and qualification) constraints affecting actions' effects; and (*iii*) nondeterminism in the initial state and in the effects of actions. We prove the correctness and completeness of the procedure, and discuss some of the issues behind its implementation and effectiveness.

1 Introduction

Propositional reasoning is a fundamental problem in many areas of Computer Science. Many researchers have put, and still put, years of effort in the design and implementation of new and more powerful SAT solvers. Most importantly, the source code of many of these implementations is freely available and can be used as the core engines of systems able to deal with more complex tasks. For example, in [Giunchiglia *et al.*, 2000] a state-of-the-art SAT solver is used as the basis for the development of 8 efficient decision procedures for classical modal logics. In [Cadoli *et al.*, 1998], a SAT solver is at the basis of a decider able to deal with Quantified Boolean Formulas (QBFs). In [Kautz and Selman, 1998], a SAT solver is used as the search engine for a very efficient planning system able to deal with STRIPS action descriptions and a completely specified initial state.

In this paper, we present a SAT-based procedure $\mathcal{C}$-SAT for planning in domains with (possibly) incomplete information about the initial state, and whose action component is specified using the highly expressive action language $\mathcal{C}$ [Giunchiglia and Lifschitz, 1998]. Thus, $\mathcal{C}$-SAT allows for planning in domains involving

- actions which can be executed concurrently;
- (ramification and/or qualification) constraints affecting actions' effects [Lin and Reiter, 1994]; and
- nondeterminism in the initial state and/or in the effects of actions.

We prove the correctness and completeness of $\mathcal{C}$-SAT, and discuss some of the issues behind its implementation and effectiveness. Finally, we present the structure of a system (called $\mathcal{C}$-PLAN) incorporating $\mathcal{C}$-SAT as core engine. Our objective in developing $\mathcal{C}$-SAT and $\mathcal{C}$-PLAN has been to see whether the good performances obtained by SAT-based planners in the classical case, would extend to more complex problems involving, e.g., concurrency and/or constraints and/or nondeterminism. Some preliminary experimental analysis that we have conducted, shows that this is indeed the case, at least for some simple nondeterministic domains [Ferraris and Giunchiglia, 2000].

The paper is thus structured as follows. In Section 2 we briefly review $\mathcal{C}$ [Giunchiglia and Lifschitz, 1998] and show a simple example that will be used throughout the whole paper. In Section 3 we state the formal results giving the grounds to the proposed procedure. In Section 4 we present the procedure and prove its correctness and completeness. In Section 5 we present

and discuss the structure of a system incorporating the proposed procedure. We end the paper with the conclusions in Section 6.

2 Action language $\mathcal{C}$

2.1 Syntax and Semantics

We start with a set of atoms partitioned into the set of *fluent symbols* and the set of *action symbols*. A *formula* is a propositional combination of atoms. An *action* is an interpretation of the action symbols. Intuitively, to execute an action α means to execute concurrently the "elementary actions" represented by the action symbols satisfying α.

An *action description* is a set of

- *static laws*, of the form:

$$\textbf{caused } F \textbf{ if } G, \tag{1}$$

- and *dynamic laws*, of the form:

$$\textbf{caused } F \textbf{ if } G \textbf{ after } H, \tag{2}$$

where F, G, H are formulas such that F and G do not contain action symbols. Both in (1) and in (2), F is called the *head* of the law.

Consider an action description D. A *state* is an interpretation of the fluent symbols that satisfies $G \supset F$ for every static law (1) in D. A *transition* is a triple $\langle \sigma, \alpha, \sigma' \rangle$ where σ, σ' are states and α is an action; intuitively σ is the initial state of the transition, and σ' is its resulting state. A formula F is *caused* in a transition $\langle \sigma, \alpha, \sigma' \rangle$ if it is

- the head of a static law (1) from D such that σ' satisfies G, or
- the head of a dynamic law (2) from D such that σ' satisfies G and $\sigma \cup \alpha$ satisfies H.

A transition $\langle \sigma, \alpha, \sigma' \rangle$ is *causally explained* in D if its resulting state σ' is the only interpretation of the fluent symbols that satisfies all formulas caused in this transition.

The *transition diagram* represented by an action description D is the directed graph which has the states of D as vertices, and which includes an edge from σ to σ' labeled α for every transition $\langle \sigma, \alpha, \sigma' \rangle$ that is causally explained in D.

2.2 An example

In rules (2), we do not write "**if** G" when G is a tautology.

Consider the following elaboration of the "safe from baby example" [Myers and Smith, 1988, Giunchiglia and Lifschitz, 1995]. There is a box and a baby crawling on the floor. The box is not dangerous for the baby if it contains dolls or if it is on the table. Otherwise, it is dangerous (e.g., because it contains hammers). We introduce the three fluent symbols *Safe*, *OnTable*, *Dolls*, and the static rules

$$\begin{array}{c} \textbf{caused } \mathit{Safe} \textbf{ if } \mathit{OnTable} \vee \mathit{Dolls}, \\ \textbf{caused } \neg \mathit{Safe} \textbf{ if } \neg \mathit{OnTable} \wedge \neg \mathit{Dolls}. \end{array} \tag{3}$$

The box can be moved by the mother or the father of the baby. The action symbols are *MPutOnTable*, *FPutOnTable*, *MPutOnFloor*, *FPutOnFloor*. The direct effects of actions is defined by the following dynamic rules:

$$\begin{array}{l} \textbf{caused } \mathit{OnTable} \textbf{ after} \\ \qquad \mathit{MPutOnTable} \vee \mathit{FPutOnTable}, \\ \textbf{caused } \neg \mathit{OnTable} \textbf{ after} \\ \qquad \mathit{MPutOnFloor} \vee \mathit{FPutOnFloor}. \end{array} \tag{4}$$

However, if the box does not contain dolls (e.g., if it is full of hammers), it can be moved on the table only by the mother and the father concurrently:

$$\begin{array}{l} \textbf{caused } \mathit{False} \textbf{ after} \\ \qquad \neg \mathit{Dolls} \wedge \neg (\mathit{MPutOnTable} \equiv \mathit{FPutOnTable}), \end{array} \tag{5}$$

where *False* represent falsity. Technically, if P is an arbitrarily chosen fluent symbol, the above rule is an abbreviation for the two rules obtained from (5) substituting first P and then $\neg P$ for *False*. All the fluents but *Safe* are inertial. This last fact, is expressed by having a pair of dynamic rules of the form

$$\begin{array}{c} \textbf{caused } P \textbf{ if } P \textbf{ after } P, \\ \textbf{caused } \neg P \textbf{ if } \neg P \textbf{ after } \neg P, \end{array} \tag{6}$$

for each fluent P different from *Safe* [McCain and Turner, 1997].

The transition diagram of the action description consisting of (3)–(6) is depicted in Figure 1. Both in the Figure and in the rest of the paper, an action α [resp. a state σ] is represented by the set of action symbols [resp. fluents] satisfied by α [resp. σ].

Consider Figure 1. As it can be observed, the transition system consists of two separated subsystems. In the first (upper in the Figure), the baby is safe no matter the location of the box. This is the case when the

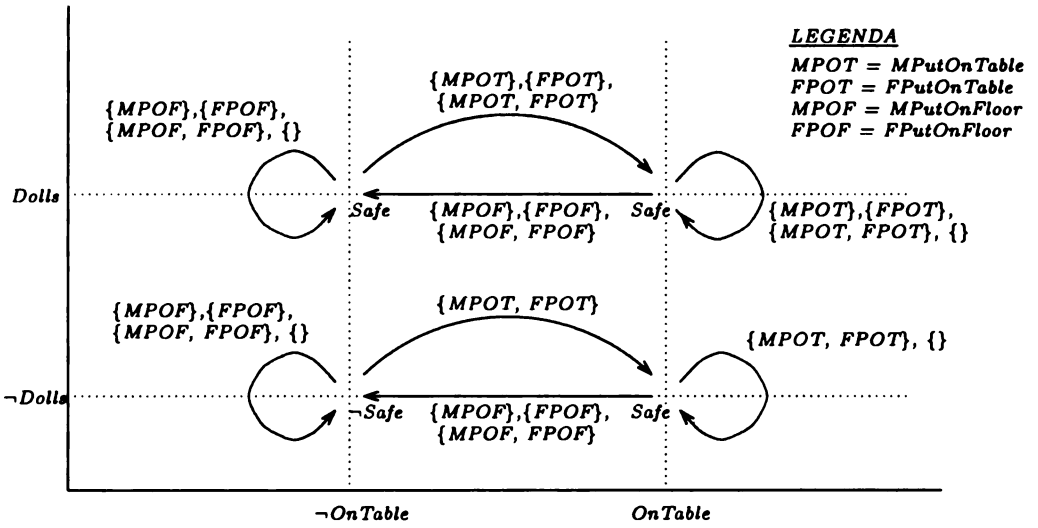


Figure 1: The transition diagram for Example 2.

box is full of dolls. In the other (lower in the Figure), the baby is safe only when the box is on the table. The example shows only few of the many expressive capabilities that $\mathcal{C}$ has. For example, the rules in (3) play the role of ramification constraints [Lin and Reiter, 1994]: moving the box on the table has the indirect effect of making the baby safe. (4) is a generalization of the traditional action preconditions from the STRIPS literature: here an action is a set of elementary actions, and *Dolls* is a precondition for the execution of $\{MPutOnTable\}$ and $\{FPutOnTable\}$. The semantics of $\mathcal{C}$ takes into account that several elementary actions can be executed concurrently. Besides this, $\mathcal{C}$ allows also, e.g., for expressing qualification constraints, fluents that change by themselves, actions with nondeterministic effects. See [Giunchiglia and Lifschitz, 1998] for more details and examples.

2.3 Computing causally explained transitions

An action description is *finite* if its signature is finite and it consists of finitely many laws. For any finite action description D, it is possible to compute a propositional formula tr_i^D whose satisfying assignments correspond to the causally explained transitions of D [Giunchiglia and Lifschitz, 1998]. To make the computation of tr_i^D easier to present, we restrict to finite action descriptions in which the head of the rules is a literal. [Giunchiglia and Lifschitz, 1998] describes how to compute tr_i^D in the general case. In this case, we say that the action description is *definite*.

Consider a definite action description D. For any number i and any formula H in the signature of of D, H_i is the expression obtained from H by substituting each atom B with B_i. Intuitively, the subscript i represents time. If P is a fluent symbol, the atom P_i expresses that P holds at time i. If A is an action symbol, the atom A_i expresses that A is among the elementary actions executed at time i. In the following, we abbreviate the static law (1) with $\langle F, G\rangle$, and the dynamic law (2) with $\langle F, G, H\rangle$. tr_i^D is the conjunction of

- for each fluent literal F, the formula

$$F_{i+1} \equiv \bigvee_{G:\langle F,G\rangle \in D} G_{i+1} \vee \bigvee_{G,H:\langle F,G,H\rangle \in D} G_{i+1} \wedge H_i,$$

- for each static causal law $\langle F, G\rangle$ in D, the formula

$$G_i \supset F_i.$$

For example, if D consists of (3)–(6), tr_i^D is equivalent to the conjunction of the formulas:

$$\begin{aligned} Dolls_{i+1} &\equiv Dolls_i, \\ Safe_i &\equiv OnTable_i \vee Dolls_i, \\ Safe_{i+1} &\equiv OnTable_{i+1} \vee Dolls_{i+1}, \\ OnTable_{i+1} &\equiv MPutOnTable_i \vee FPutOnTable_i \vee \\ & \quad OnTable_i \wedge \neg MPutOnFloor_i \wedge \neg FPutOnFloor_i, \\ (MPutOnTable_i &\vee FPutOnTable_i) \supset \\ & \quad \neg(MPutOnFloor_i \vee FPutOnFloor_i), \\ \neg Dolls_i &\supset (MPutOnTable_i \equiv FPutOnTable_i). \end{aligned} \tag{7}$$

In the rest of the paper, tr_i^D is the formula defined as above if D is definite, and the formula defined analogously to $ct_1(D)$ in [Giunchiglia and Lifschitz, 1998], otherwise. Furthermore, for any interpretation

μ and natural number i, μ_i abbreviates the expression $\bigwedge_{L:L \text{ is a literal}, \mu \models L} L_i$.

Proposition 1 ([Giunchiglia and Lifschitz, 1998]) *Let D be a finite action description. Let $\langle \sigma, \alpha, \sigma' \rangle$ be a transition of D. $\langle \sigma, \alpha, \sigma' \rangle$ is causally explained in D iff $\sigma_i \wedge \alpha_i \wedge \sigma'_{i+1}$ entails tr_i^D.*

3 Formal results

A *history* for an action description D is a path in the corresponding transition diagram, that is, a finite sequence

$$\sigma^0, \alpha^1, \sigma^1, \ldots, \alpha^n, \sigma^n \tag{8}$$

($n \geq 0$) such that $\sigma^0, \sigma^1, \ldots, \sigma^n$ are states, $\alpha^1, \ldots, \alpha^n$ are actions, and

$$\langle \sigma^{i-1}, \alpha^i, \sigma^i \rangle \qquad (1 \leq i \leq n)$$

are causally explained transitions of D.

Let D be a finite action description. A *planning problem* for D is characterized by two formulas I and G in the fluent signature, i.e., is a triple $\langle I, D, G \rangle$. A state σ is *initial* [resp. *goal*] if σ satisfies I [resp. G]. A *plan* is a finite sequence $\alpha^1; \ldots; \alpha^n$ ($n \geq 0$) of actions.

3.1 Possible plans

Consider a planning problem $\pi = \langle I, D, G \rangle$. A plan $\alpha^1; \ldots; \alpha^n$ is *possible* for π if there exists a history (8) for D such that σ^0 is an initial state, and σ^n is a goal state. For example, the plan consisting of the empty sequence of actions is possible for the planning problem[1]

$$\langle \mathit{True}, (3)\text{–}(6), \mathit{Safe} \rangle. \tag{9}$$

In fact, there exists an initial state which is also a goal state.

The following Theorem is similar to Proposition 2 in [McCain and Turner, 1998].

Theorem 1 *Let $\pi = \langle I, D, G \rangle$ be a planning problem. A plan $\alpha^1; \ldots; \alpha^n$ is possible for π iff the formula*

$$\bigwedge_{i=0}^{n-1} \alpha_i^{i+1} \wedge I_0 \wedge \bigwedge_{i=0}^{n-1} tr_i^D \wedge G_n$$

is satisfiable.

The execution of a possible plan is not ensured to lead to a goal state. Indeed, if D is deterministic (i.e., if for any state σ and action α of D, D has at most one causally explained transition $\langle \sigma, \alpha, \sigma' \rangle$), and there is only one initial state, then executing a possible plan leads to a goal state. This is the idea underlying planning as satisfiability in [Kautz and Selman, 1992]. However, this is not always the case in our setting, where actions can be nondeterministic and there can be multiple initial states. For example, considering the planning problem (9), executing the possible plan consisting of the empty sequence of actions is not ensured to lead to a goal state: In fact, there is an initial state which is not a goal state.

3.2 Valid Plans

As pointed out in [McCain and Turner, 1998], in order to be sure that a plan $\alpha^1; \ldots; \alpha^n$ is good (they say "valid"), it is not enough to check that for any history (8) such that σ^0 is an initial state, σ^n is a goal state. According to this definition, the plan $\{\mathit{MPutOnTable}\}$ would be valid for the planning problem (9). Indeed, $\{\mathit{MPutOnTable}\}$ is not valid since this action is not "executable" in the initial states satisfying $\neg \mathit{Dolls}$. Intuitively, we have to check that the plan is also "always executable" in any initial state, i.e., executable for any initial state and any possible outcome of the actions in the plan. To make the notion of valid plan precise, we need the following definitions.

Consider a finite action description D and a plan $\vec{\alpha} = \alpha^1; \ldots; \alpha^n$.

An action α is *executable* in a state σ if for some state σ', $\langle \sigma, \alpha, \sigma' \rangle$ is a causally explained transition of D. Let σ^0 be a state. The plan $\vec{\alpha}$ is *always executable in* σ^0 if for any history

$$\sigma^0, \alpha^1, \sigma^1, \ldots, \alpha^k, \sigma^k$$

with $k < n$, α^{k+1} is executable in σ^k. For example, if D consists of (3)–(6), the plan $\{\mathit{MPutOnFloor}\}; \{\mathit{FPutOnFloor}\}$ is always executable in any state, while the plan $\{\mathit{MPutOnFloor}\}; \{\mathit{FPutOnTable}\}$ is always executable only in the states satisfying *Dolls*.

Assume that $\vec{\alpha}$ is a plan which is always executable in a state σ^0. A state σ^n is a *possible result of executing $\vec{\alpha}$ in σ^0* if there exists an history (8) for D. For example, in the case of (3)–(6), the state $\{\}$ (i.e., the state which does not satisfy any fluent symbol) is a possible result of executing $\{\mathit{MPutOnFloor}\}; \{\mathit{FPutOnFloor}\}$ in any state satisfying $\neg \mathit{Dolls}$.

Let $\pi = \langle I, D, G \rangle$ be a planning problem. A plan $\vec{\alpha} = \alpha^1; \ldots; \alpha^n$ is *valid* for π if for any initial state σ^0,

- $\vec{\alpha}$ is always executable in σ^0, and

[1] *True* is an abbreviation for a fixed tautology.

- any possible result of executing $\vec{\alpha}$ in σ^0 is a goal state.

Considering the planning problem (9), $\{MPutOnTable\}$ is not valid, while $\{MPutOnTable, FPutOnTable\}$ is valid.

Proposition 2 *Let $\pi = \langle I, D, G\rangle$ be a planning problem. Let $\vec{\alpha} = \alpha^1; \ldots; \alpha^n$ be a plan which is always executable in any initial state. $\vec{\alpha}$ is valid for π iff*

$$I_0 \wedge \wedge_{i=0}^{n-1} \alpha_i^{i+1} \wedge \wedge_{i=0}^{n-1} tr_i^D \models G_n.$$

Thus, if a plan $\vec{\alpha}$ is always executable in any initial state, Proposition 2 establishes a necessary and sufficient condition for determining whether $\vec{\alpha}$ is valid. The following Proposition, allows us to determine when an action is executable in a state. Let $Poss_i^D$ be the formula

$$\exists p^1 \ldots \exists p^n tr_i^D[P_{i+1}^1/p^1, \ldots, P_{i+1}^n/p^n] \qquad (10)$$

where $P^1, \ldots, P^n$ are all the fluent symbols in D, and $tr_i^D[P_{i+1}^1/p^1, \ldots, P_{i+1}^n/p^n]$ denotes the formula obtained from tr_i^D by substituting each fluent P_{i+1}^k with a distinct propositional variable p^k. Notice that the propositional variables and the bounding quantifiers can always be eliminated, although the formula can become much longer in the process.[2] For example, if tr_i^D is the conjunction of the formulas (7), then $Poss_i^D$ is equivalent to the conjunction of the following formulas:

$$\begin{aligned} &Safe_i \equiv OnTable_i \vee Dolls_i, \\ &(MPutOnTable_i \vee FPutOnTable_i) \supset \\ &\qquad \neg(MPutOnFloor_i \vee FPutOnFloor_i), \\ &\neg Dolls_i \supset (MPutOnTable_i \equiv FPutOnTable_i). \end{aligned} \qquad (11)$$

Proposition 3 *Let α be an action and let σ be a state of a finite action description D. α is executable in σ iff $\sigma_i \wedge \alpha_i$ entails $Poss_i^D$.*

The above Proposition establishes, e.g., that in the action description (3)–(6), the actions $\{MPutOnTable\}$ and $\{FPutOnTable\}$ are not executable in states satisfying $\neg Dolls$, as it can be easily checked from (11).

Let trt_i^D be the formula

$$(tr_i^D \wedge \neg Z_i \wedge \neg Z_{i+1}) \vee ((Z_i \vee \neg Poss_i^D) \wedge Z_{i+1}), \qquad (12)$$

where Z is a newly introduced fluent symbol. Given that $tr_i^D \supset Poss_i^D$ holds, (12) is equivalent to

$$(\neg Z_i \vee Z_{i+1}) \wedge (tr_i^D \vee Z_{i+1}) \wedge (\neg Poss_i^D \vee Z_i \vee \neg Z_{i+1}).$$

Intuitively, if s^f is the fluent signature of D, trt_i^D determines (in the sense of Proposition 1) the transition relation of an automaton

- whose set of states corresponds to the set of assignments of the signature $s^f \cup \{Z\}$, and
- whose transitions are labeled with the actions of D and are such that there is a transition from a state σ to a state σ' with label α if and only if
  - σ and σ' satisfy $\neg Z$ and $\langle \sigma_D, \alpha, \sigma'_D\rangle$ is a causally explained transition of D, or
  - σ satisfies $\neg Z$, σ' satisfies Z and α is not executable in σ_D, or
  - σ and σ' satisfy Z,

  where σ_D is the restriction of σ to s^f (and similarly for σ'_D).

The above intuition is made precise by the following Proposition.

Proposition 4 *Let D be a finite action description. Let σ, σ' be two states of D, and let α be an action. The following three facts hold:*

1. *$\langle \sigma, \alpha, \sigma'\rangle$ is a causally explained interpretation of D iff $\sigma_i \wedge \neg Z_i \wedge \alpha_i \wedge \sigma'_{i+1} \wedge \neg Z_{i+1}$ entails (12).*
2. *α is not executable in σ iff $\sigma_i \wedge \neg Z_i \wedge \alpha_i \wedge Z_{i+1}$ entails (12).*
3. *(12) entails $Z_i \supset Z_{i+1}$.*

The following Theorem follows from Propositions 2 and 4. $State_0^D$ is the formula

$$\bigwedge_{F,G:\langle F,G\rangle \in D} G_0 \supset F_0,$$

representing the set of "possible initial states". If D is (3)–(6), then $State_0^D$ is

$$Safe_0 \equiv OnTable_0 \vee Dolls_0.$$

Theorem 2 *Let D be a finite action description. A plan $\alpha^1; \ldots; \alpha^n$ is valid for a planning problem $\langle I, D, G\rangle$ iff*

$$I_0 \wedge State_0^D \wedge \neg Z_0 \wedge \wedge_{i=0}^{n-1} \alpha_i^{i+1} \wedge \wedge_{i=0}^{n-1} trt_i^D \models G_n \wedge \neg Z_n.$$

[2] If for each action we have its preconditions explicitly listed (as, e.g., in STRIPS), it is possible to compute the propositional formula equivalent to (10) by simple syntactic manipulations (see, e.g., [Ferraris and Giunchiglia, 2000]).

Considering the planning problem (9), the above Theorem can be used, e.g., to establish that the plan $\{MPutOnTable\}$ is not valid, while $\{MPutOnTable, FPutOnTable\}$ is valid. To see why, consider the formula

$$I_0 \wedge State_0^D \wedge \neg Z_0 \wedge \wedge_{i=0}^{n-1} \alpha_i^{i+1} \wedge \wedge_{i=0}^{n-1} trt_i^D. \quad (13)$$

In both cases $n = 1$. But,

- When $\alpha^1 = \{MPutOnTable\}$, (13) is equivalent to the conjunction of the formulas

$$Safe_0 \equiv OnTable_0 \vee Dolls_0,$$
$$\neg Z_0,$$
$$MPutOnTable \wedge \neg FPutOnTable$$
$$\wedge \neg MPutOnFloor \wedge \neg FPutOnFloor,$$

  and the formulas

$$Safe_1 \vee Z_1,$$
$$Dolls_0 \vee Z_1,$$
$$Dolls_1 \vee Z_1,$$
$$OnTable_1 \vee Z_1,$$
$$\neg Dolls_0 \vee \neg Z_1.$$

  This conjunction does not entail $Safe_1 \wedge \neg Z_1$. Indeed, $\{MPutOnTable\}$ is a valid plan for (9) if *Dolls* initially holds.

- When $\alpha^1 = \{MPutOnTable, FPutOnTable\}$, (13) is equivalent to the conjunction of

$$Safe_0 \equiv OnTable_0 \vee Dolls_0,$$
$$\neg Z_0,$$
$$MPutOnTable \wedge FPutOnTable$$
$$\wedge \neg MPutOnFloor \wedge \neg FPutOnFloor,$$

  and the formulas

$$Safe_1,$$
$$OnTable_1,$$
$$Dolls_1 \equiv Dolls_0,$$
$$\neg Z_1.$$

  This conjunction obviously entails $Safe_1 \wedge \neg Z_1$.

4 Computing valid plans in $\mathcal{C}$

Consider a planning problem $\pi = \langle I, D, G \rangle$. Thanks to Theorem 2, we may divide the problem of finding a valid plan for π into two parts:

1. *generate* a (possible) plan, and
2. *test* whether the generated plan is also valid.

The testing phase can be performed using any state-of-the-art complete SAT solver. According to Theorem 2, a plan $\alpha^1; \ldots; \alpha^n$ is valid if and only if

$$I_0 \wedge State_0^D \wedge \neg Z_0 \wedge \wedge_{i=0}^{n-1} \alpha_i^{i+1} \wedge \wedge_{i=0}^{n-1} trt_i^D \wedge \neg(G_n \wedge \neg Z_n) \quad (14)$$

is not satisfiable. For the generation phase, different strategies can be used:

1. From the generation of arbitrary plans of length n;
2. Through the generation of a subset of the possible plans of length n;
3. To the generation of the whole set of possible plans of length n.

By checking the validity of each generated plan, we obtain a correct but possibly incomplete procedure in the first two cases; and a correct and complete procedure in the last case. Given a planning problem π and a natural number n, we say that a procedure is

- *correct* (for π, n) if any returned plan $\alpha_1; \ldots; \alpha_n$ is valid for π, and
- *complete* (for π, n) if it returns *False* when there is no valid plan $\alpha_1; \ldots; \alpha_n$ for π.

By Theorem 1, possible plans of length n can be generated by finding assignments satisfying the formula

$$I_0 \wedge \wedge_{i=0}^{n-1} tr_i^D \wedge G_n. \quad (15)$$

This can be accomplished using incomplete SAT solvers like GSAT [Selman *et al.*, 1992], or complete SAT solvers like SATO [Zhang, 1997]. For the generation of the whole set of possible plans, we may use a complete SAT solver, and

- at step 0, ask for an assignment satisfying (15), and
- at step $i+1$, ask for an assignment satisfying (15) and the negation of the plans generated in the previous steps,

till no more satisfying assignments are found. This method for generating all possible plans has the advantage that the SAT decider is used as a blackbox. The obvious disadvantage is that the size of the input formula checked by the SAT solver may become exponentially bigger than the original one. A better solution is to invoke the test *inside* the SAT procedure whenever

$P := I_0 \wedge \wedge_{i=0}^{n-1} tr_i^D \wedge G_n$;
$V := I_0 \wedge State_0^D \wedge \neg Z_0 \wedge \wedge_{i=0}^{n-1} trt_i^D \wedge \neg(G_n \wedge \neg Z_n)$;

function $\mathcal{C}$-SAT()
  return $\mathcal{C}$-SAT_GEN$_{DP}$(*cnf*(P), {}).

function $\mathcal{C}$-SAT_GEN$_{DP}(\varphi, \mu)$
  if $\varphi = \{\}$ **then return** $\mathcal{C}$-SAT_TEST(μ); /* base */
  if $\{\} \in \varphi$ **then return** *False*; /* backtrack */
  if { a unit clause $\{L\}$ occurs in φ } /* unit */
    then return $\mathcal{C}$-SAT_GEN$_{DP}$(*assign*$(L, \varphi), \mu \cup \{L\})$;
  $L :=$ { a literal occurring in φ };
  return $\mathcal{C}$-SAT_GEN$_{DP}$(*assign*$(L, \varphi), \mu \cup \{L\})$ **or** /* split */
      $\mathcal{C}$-SAT_GEN$_{DP}$(*assign*$(\overline{L}, \varphi), \mu \cup \{\neg A\})$.

function $\mathcal{C}$-SAT_TEST(μ)
  $\alpha :=$ { the set of literals in μ corresponding to action literals};
  foreach {plan $\alpha^1; \ldots; \alpha^n$ s.t. each element in α is a conjunct in $\wedge_{i=0}^{n-1} \alpha_i^{i+1}$}
    if not SAT$(\wedge_{i=0}^{n-1} \alpha_i^{i+1} \wedge V)$ **then exit with** $\alpha^1; \ldots; \alpha^n$;
  return *False*.

Figure 2: $\mathcal{C}$-SAT and $\mathcal{C}$-SAT_GEN$_{DP}$

a possible plan is found. In the case of the Davis-Putnam (DP) procedure [Davis and Putnam, 1960, Davis *et al.*, 1962], we get the procedure $\mathcal{C}$-SAT represented in Figure 2. In the Figure,

- *cnf*(P) is a set of clauses corresponding to P. The transformation from a formula into a set of clauses can be performed using the conversions based on "renaming" (see, e.g., [Tseitin, 1970, Plaisted and Greenbaum, 1986]).
- $\overline{L}$ is the literal complementary to L.
- For any literal L and set of clauses φ, *assign*(L, φ) is the set of clauses obtained from φ by
  - deleting the clauses in which L occurs as a disjunct, and
  - eliminating $\overline{L}$ from the others.

Main Theorem *Let $\pi = \langle I, D, G \rangle$ be a planning problem. Let n be a natural number. $\mathcal{C}$-SAT is correct and complete for π, n.*

5 Implementation

We have implemented a system (called $\mathcal{C}$-PLAN) implementing the above ideas. The architecture of $\mathcal{C}$-PLAN is represented in Figure 3. In the Figure, blocks stand for routines of the system, and arrows show the data flow. The input/output behavior of each module is specified by the labels in the input and output arrows (for the meaning of the labels see also Figure 2). With reference to the Figure, the following observations are worthwhile.

CCALC computes the set of clauses corresponding to the transition relation tr_0^D of any (not necessarily definite) action description D.[3] *CTCALC* adopts some of the techniques described in [Cadoli *et al.*, 1998] for computing a propositional formula equivalent to (10). For any planning problem, *CCALC* and *CTCALC* are executed only once. Of course, in a simplified setting in which action preconditions' are explicitly listed (as in [Ferraris and Giunchiglia, 2000, Smith and Weld, 1998]) a propositional formula equivalent to (10) could be computed by means of simple syntactic manipulations of the input action description.

$\mathcal{C}$-SAT and $\mathcal{C}$-SAT_TEST are implemented on top of the *SAT system [Tacchella, 1999, Giunchiglia *et al.*, 2000].[4] We expect to obtain better performances by imposing that

[3] The *CCALC* module has been kindly provided by Norman McCain and is part of the *CCALC* system, see `http://www.cs.utexas.edu/users/tag/cc`.

[4] *SAT is a platform built on top of SATO ver. 3.2 [Zhang, 1997], for the development of SAT-based decision procedures. See `http://www.mrg.dist.unige.it/~tac/StarSAT.html`.

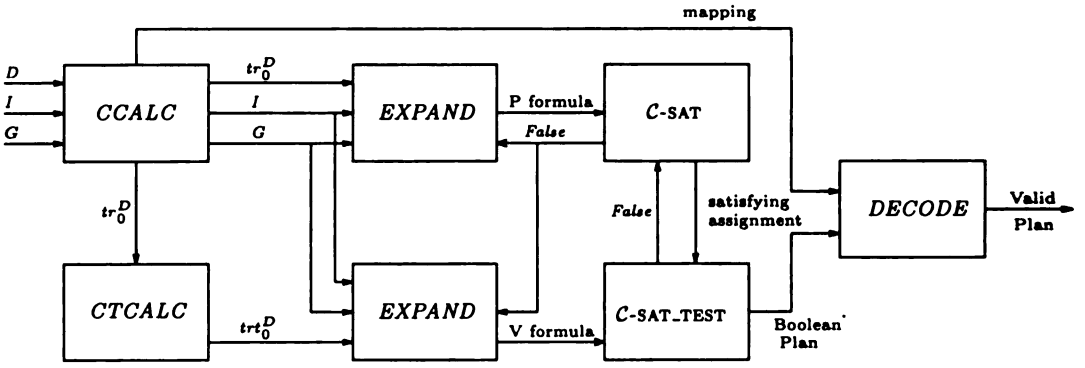


Figure 3: Architecture of the system.

1. C-SAT_GEN$_{DP}$ first splits on variables corresponding to actions (as in [Giunchiglia *et al.*, 1998]), and
2. once all the variables for actions are assigned, the corresponding plan is tested for validity without seeing whether it is possible or not.

In this mode, a plan will be generated and tested at most once. Otherwise, this may not be the case. Consider, e.g., the planning problem $\langle True, (3)$–$(6), OnTable \vee Dolls \rangle$. For $n = 0$, P is $OnTable_0 \vee Dolls_0$ and C-SAT_GEN$_{DP}(P, \{\})$ will generate two assignments satisfying P. As a consequence, the empty sequence of actions will be tested twice for validity.

EXPAND is a simple module generating P/V formulas with a bigger value for n at each step. Notice that, in C, the nonexistence of a valid plan of length n does not ensure the nonexistence of a valid plan of length $m < n$. In fact, as we said in Section 2, C allows for fluents that change by themselves, i.e., without executing any elementary action (see, e.g., the "pendulum domain" in [Giunchiglia and Lifschitz, 1998]). The simple solution is thus to consider $n = 0, 1, 2, 3, 4, \ldots$. If we want to test for $n = 0, 1, 2, 4, 8, \ldots$, and, at each step, check whether a valid plan of length $m < n$ exists, a simple solution is to

1. add an action *NoOp* to the action signature, and
2. consider as the "new" transition relation of D, the formula

   $$(tr_i^D \wedge \neg NoOp_i) \vee ((\wedge_{P \in s^f} P_{i+1} \equiv P_i) \wedge NoOp_i)$$

   where s^f is the fluent signature of D.

Finally, our objective in developing C-SAT and C-PLAN has been to see whether the good performances obtained by SAT-based planners in the classical case, would extend to more complex problems involving, e.g., concurrency and/or constraints and/or nondeterminism. About C-SAT performances, we expect that most of its computation time will be spent by C-SAT_GEN$_{DP}$. In particular, we expect validity tests not to be onerous. Indeed, when checking whether (14) is satisfiable, each variable corresponding to an action gets assigned by unit propagation at the beginning of the search. Consequently, many other fluent variables may get assigned by unit propagation before the search starts. We have performed some tests focusing on "simple" nondeterministic domains. Intuitively, a domain is "simple", if

- there are no static laws;
- concurrency is not allowed; and
- each action A is characterized by $n + 2$ ($n \geq 1$) finite sets of fluents $P, E, N_1, \ldots, N_n$: P and E list respectively A's preconditions and effects as in STRIPS, while each N_i represents one of the possible indeterminate outcomes of A.

For "simple" nondeterministic domains, the theory and procedures that we have presented for C can be simplified and extended. For example, "regular parallel" or "simple-split sequential" encodings in the style of [Kautz and Selman, 1996, Kautz *et al.*, 1997, Ernst *et al.*, 1997] are possible. Furthermore, if we restrict to "simple" nondeterministic domains, we can perform some comparative analysis with other planners. The tests show that

- $\mathcal{C}$-SAT performances are poor when there are a huge number of possible plans and only few of them are valid, but
- some simple heuristics can be devised that greatly reduce the number of generated possible plans.
- The result is that $\mathcal{C}$-SAT —at least for simple domains— can be competitive with CGP [Smith and Weld, 1998]. CGP is based on planning graphs [Blum and Furst, 1995], and outperforms BURIDAN [Kushmerick *et al.*, 1995] and UDTPOP [Peot, 1998] by orders of magnitude (see [Smith and Weld, 1998]).

See [Ferraris and Giunchiglia, 2000] for more details.

6 Conclusions and future work

We have presented a SAT-based procedure capable to deal with planning domains having incomplete information about the initial state, and whose underlying transition system is specified in $\mathcal{C}$. $\mathcal{C}$ allows for, e.g., concurrency, constraints, and nondeterminism.

This work can be seen as a follow up of [McCain and Turner, 1998]. In that paper, the language of causal theories is considered, and the notions of possible and valid plans are introduced. The action language $\mathcal{C}$ is based on [McCain and Turner, 1997], and is less expressive than the language of causal theories used in [McCain and Turner, 1998]. However, the focus in [McCain and Turner, 1998] is on stating some conditions under which a possible plan is also valid. No procedure for computing valid plans in the general case (e.g., with multiple initial states or actions with nondeterministic effects) is given.

There is a huge literature in planning with incomplete information. The approaches can be divided according to the type of plan returned, either conditional (where the sequence of actions being executed depends on some sensory information dynamically acquired) or conformant (where a plan is a sequence of actions which, if executed, is ensured to achieve the goal, no matter the nondeterminism in the given planning problem). If we restrict to the literature in conformant planning, the latest works are [Smith and Weld, 1998, Cimatti and Roveri, 1999]. In such approaches the action language used misses some of the $\mathcal{C}$ expressive capabilities (e.g., concurrency, qualification constraints).

Interestingly, [Baral *et al.*, 1999] shows that feasible planning with incomplete information about the initial state is $\Sigma_2\mathbf{P}$ complete (assuming that there are more than two action symbols) even using the simple $\mathcal{A}$ action language [Gelfond and Lifschitz, 1993]. Intuitively, a plan is "feasible" if its length is bounded by a polynomial in the length of the input planning problem. The idea of the proof showing that the problem belongs to $\Sigma_2\mathbf{P}$, is that we may first guess a plan and then test whether the plan is good. Indeed, this is the idea underlying our procedure.

Finally, [Rintanen, 1999] shows that finding conditional plans of polynomial size (i) is in $\Sigma_2\mathbf{P}$, and (ii) does not belong to $\mathbf{NP}$. Given that it is not feasible to encode conditional planning into SAT, he uses the more expressive language of Quantified Boolean Formulas as target. Indeed, we are not encoding problems into SAT as in, e.g., [Kautz and Selman, 1992]. We use a SAT checker as the basis of our procedure $\mathcal{C}$-SAT.

Acknowledgements

Paolo Ferraris, Vladimir Lifschitz, Jussi Rintanen and Hudson Turner are thanked for comments on an earlier version of this paper. Paolo Ferraris has also participated to the design of the architecture of $\mathcal{C}$-PLAN. Norman McCain has made possible to integrate *CCALC* in $\mathcal{C}$-PLAN. The author is supported by ASI, CNR and MURST.

References

[Baral *et al.*, 1999] Chitta Baral, Vladik Kreinovich, and Raúl Trejo. Computational complexity of planning and approximate planning in presence of incompleteness. In *Proc. IJCAI-99*, pages 948–953, 1999.

[Blum and Furst, 1995] Avrim Blum and Merrick Furst. Fast planning through planning graph analysis. In *Proc. of IJCAI-95*, pages 1636–1642, 1995.

[Cadoli *et al.*, 1998] M. Cadoli, A. Giovanardi, and M. Schaerf. An algorithm to evaluate quantified boolean formulae. In *Proc. AAAI*, 1998.

[Cimatti and Roveri, 1999] Alessandro Cimatti and Marco Roveri. Conformant planning via model checking. In *Lecture Notes in Computer Science, Proc. of the 5th European Conference on Planning (ECP-99)*. Springer, September 1999.

[Davis and Putnam, 1960] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.

[Davis *et al.*, 1962] M. Davis, G. Longemann, and D. Loveland. A machine program for theorem proving. *Journal of the ACM*, 5(7), 1962.

[Ernst *et al.*, 1997] Michael Ernst, Todd Millstein, and Daniel Weld. Automatic SAT-compilation of planning problems. In *Proc. IJCAI-97*, 1997.

[Ferraris and Giunchiglia, 2000] Paolo Ferraris and Enrico Giunchiglia. Planning as satisfiability in simple nondeterministic domains, 2000. Manuscript.

[Gelfond and Lifschitz, 1993] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–322, 1993.

[Giunchiglia and Lifschitz, 1995] Enrico Giunchiglia and Vladimir Lifschitz. Dependent fluents. In *Proc. IJCAI-95*, pages 1964–1969, 1995.

[Giunchiglia and Lifschitz, 1998] Enrico Giunchiglia and Vladimir Lifschitz. An action language based on causal explanation: Preliminary report. In *Proc. AAAI-98*, pages 623–630, 1998.

[Giunchiglia *et al.*, 1998] E. Giunchiglia, A. Massarotto, and R. Sebastiani. Act, and the rest will follow: Exploiting determinism in planning as satisfiability. In *Proc. AAAI*, 1998.

[Giunchiglia *et al.*, 2000] E. Giunchiglia, F. Giunchiglia, and A. Tacchella. SAT-Based Decision Procedures for Classical Modal Logics. *Journal of Automated Reasoning*, 2000. To appear.

[Kautz and Selman, 1992] Henry Kautz and Bart Selman. Planning as satisfiability. In *Proc. ECAI-92*, pages 359–363, 1992.

[Kautz and Selman, 1996] Henry Kautz and Bart Selman. Pushing the envelope: planning, propositional logic and stochastic search. In *Proc. AAAI-96*, pages 1194–1201, 1996.

[Kautz and Selman, 1998] Henry Kautz and Bart Selman. BLACKBOX: A new approach to the application of theorem proving to problem solving. In *Working notes of the Workshop on Planning as Combinatorial Search, held in conjunction with AIPS-98*, 1998.

[Kautz *et al.*, 1997] Henry Kautz, David McAllester, and Bart Selman. Encoding plans in propositional logic. In *Proc. KR-96*, pages 374–384, 1997.

[Kushmerick *et al.*, 1995] N. Kushmerick, S. Hanks, and D. Weld. An algorithm for probabilistic planning. *Artificial Intelligence*, 76(1-2):239–286, 1995.

[Lin and Reiter, 1994] Fangzhen Lin and Raymond Reiter. State constraints revisited. *Journal of Logic and Computation*, 4:655–678, 1994.

[McCain and Turner, 1997] Norman McCain and Hudson Turner. Causal theories of action and change. In *Proc. AAAI-97*, pages 460–465, 1997.

[McCain and Turner, 1998] Norman McCain and Hudson Turner. Fast satisfiability planning with causal theories. In *Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, 1998.

[Myers and Smith, 1988] Karen Myers and David Smith. The persistence of derived information. In *Proc. AAAI-88*, pages 496–500, 1988.

[Peot, 1998] M. Peot. *Decision-Theoretic Planning.* PhD thesis, Stanford University, Dept. of Engineering-Economic Systems, 1998.

[Plaisted and Greenbaum, 1986] D.A. Plaisted and S. Greenbaum. A Structure-preserving Clause Form Translation. *Journal of Symbolic Computation*, 2:293–304, 1986.

[Rintanen, 1999] Jussi Rintanen. Constructing conditional plans by a theorem prover. *Journal of Artificial Intelligence Research*, 10:323–352, 1999.

[Selman *et al.*, 1992] B. Selman, H. Levesque., and D. Mitchell. A New Method for Solving Hard Satisfiability Problems. In *Proc. of the 10th National Conference on Artificial Intelligence*, pages 440–446, 1992.

[Siekmann and Wrightson, 1983] Jörg Siekmann and Graham Wrightson, editors. *Automation of Reasoning: Classical Papers in Computational Logic 1967–1970*, volume 2. Springer-Verlag, 1983.

[Smith and Weld, 1998] David Smith and Daniel Weld. Conformant graphplan. In *Proc. AAAI-98*, pages 889–896, 1998.

[Tacchella, 1999] Armando Tacchella. *SAT system description. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Collected Papers from the International Description Logics Workshop (DL'99)*. CEUR, July 1999.

[Tseitin, 1970] G. Tseitin. On the complexity of proofs in propositional logics. *Seminars in Mathematics*, 8, 1970. Reprinted in [Siekmann and Wrightson, 1983].

[Zhang, 1997] H. Zhang. SATO: An efficient propositional prover. In William McCune, editor, *Proceedings of the 14th International Conference on Automated deduction*, volume 1249 of *LNAI*, pages 272–275, Berlin, July13–17 1997. Springer.

Learning generalized policies in planning using concept languages

Mario Martín
Dept. de Llenguatges i Sistemes Inf.
Universitat Politècnica de Catalunya
08034 Barcelona, Spain

Héctor Geffner
Departamento de Computación y TI
Universidad Simón Bolívar
Aptdo. 89000, Caracas, Venezuela

Abstract

In this paper we are concerned with the problem of learning how to solve planning problems in one domain given a number of solved instances. This problem is formulated as the problem of inferring a function that operates over all instances in the domain and maps states and goals into actions. We call such functions *generalized policies* and the question that we address is how to learn suitable representations of generalized policies from data. This question has been addressed recently by Roni Khardon [16]. Khardon represents generalized policies using an ordered list of existentially quantified rules that are inferred from a training set using a version of Rivest's learning algorithm [22]. Here, we follow Khardon's approach but represent generalized policies in a different way using a *concept language*. We show through a number of experiments in the blocks-world that the concept language yields a better policy using a smaller set of examples and no background knowledge. The policy representation is related to the indexical-functional representations advocated by Agre and Chapman [1] and domain concepts such as 'the-next-needed-block' and 'a-well-placed-block' are identified from scratch.

1 Introduction

Planning is an essential part of intelligent behavior. In AI, a planner is given an initial state and a goal, and finds a sequence of actions that maps the state into the goal [23]. This problem has been tackled by a number of algorithms and in recent years substantial progress has been made [6, 15]. Still the problem is computationally hard and the best algorithms are bound to fail on certain classes of instances [10]. An alternative that has been proposed is to use knowledge of the planning domain for guiding the planning process (e.g., [3]). In the blocks world, for example, one may want to say 'never to move a block A that is well-placed' where the predicate 'well-placed' is defined in a suitable way. Planners that rely on domain-dependent control knowledge can outperform the best domain-independent planners [3, 11] but such knowledge is not always easy to provide.

In this paper, we are concerned with the problem of *learning* domain-dependent control knowledge. More precisely, we will be interested in the problem of learning how to solve a problem in a domain, given solutions to a number of small instances. This problem is formulated as the problem of inferring a function that operates over all instances in the domain and maps states and goals into actions. We call such functions *generalized policies* in contrast with the policies used in dynamic programming that have a more limited scope. A generalized policy for the blocks-world may say things like 'pick up a misplaced block if clear', 'put current block on destination if destination block is clear and well placed', etc. The question that we address is how to learn suitable representations of such policies from a number of solved instances. This question has been addressed recently by Roni Khardon [16]. Khardon represents generalized policies using an ordered list of existentially quantified rules and infers such representations from data using a version of Rivest's learning algorithm [22]. Khardon's results are encouraging and show that the learned policies can solve problems that the planner used as teacher cannot solve. The approach, however, has also some weaknesses. First, it relies on background knowledge in the form of support predicates that express key features of the domain, and second, the resulting policies do not generalize so well. In this paper we aim to show that these weaknesses

can be addressed by learning generalized policies expressed using a *concept language* [8, 9, 12, 19]. Concept languages have the expressive power of fragments of standard first-order logic but with a *syntax* that is suited for representing and reasoning with *classes* of objects. For example, the class of 'well-placed' objects in the blocks world domain can be defined in a very compact way in terms of the primitive blocks world predicates such as *on*, *clear*, etc. In this paper, we aim to show that this convenience also makes it simpler to *learn* such concepts while simultaneously learning the policies based on them.

More precisely, through a number of experiments in the blocks world we obtain that

- the use of a different representation language has a significant impact in the effectiveness and generality of the approach; in particular, the concept language yields a better policy using less examples and *no background knowledge*
- the algorithm learns useful concepts and rules simultaneously, identifying domain concepts such as 'the-next-needed-block', 'a-well-placed-block', etc. from scratch
- the generalized policies using these learned concepts are closely related to the indexical-functional representations for action selection advocated by Agre and Chapman [1].

We also discuss current and inherent limitations of the approach and future work.

2 Policies

A problem in classical planning is given by a set of actions, an initial state, and a goal. These problems can be formulated as problems of search from the initial state to the set of goal states by applying actions that map one state into another. The result of this search is a path in state-space or a plan [20]. A related state-space formulation can be obtained by viewing planning from the perspective of dynamic programming [5]. In a dynamic programming formulation, the solutions of the planning problem is not an *action sequence* but a *policy* π that maps states into actions. The plan $a_0, a_1, \ldots, a_n$ that achieves the goal from an initial state s_0 can be obtained by setting a_0 to the action $\pi(s_0)$, a_1 to the action $\pi(s_1)$, and so on, where s_{i+1} is the result of applying action a_i in state s_i. Note that while a *plan* encodes the solution of a *single* planning instance, a *policy* encodes the solution of a *class* of instances; namely, the instances that differ from the original instance in the initial state.

Using the same ideas it is possible to represent solutions of larger classes of problems by defining suitable policies. For example, a policy defined over all states and all goals can encode the solutions of the class of problems that differ on both the initial state and the goal. Such policies map states and goals into actions.

In this paper, we are interested in finding the solution to a still larger class of problems. We are interested in learning how to solve *any* problem instance in a given domain. The type of policies needed to solve such large class of instances can be made precise by making explicit the description of planning problems. A planning problem is normally made up of two parts:

- a *domain description* in the form of some action *schemas*, and
- an *instance description* in the form of a set of object names, a state, and a goal

This distinction is explicit in the PDDL language, a recent standard for stating planning problems [18].

Given this distinction, a general policy for solving *all* problems in the domain must take an *instance description* as input, and map it into an action. We call such policies defined over all the domain instances, *generalized policies*.

The algorithms used for computing policies over the state-space cannot be generalized easily to the space of all instances. However, while no general method is known for computing such generalized policies, such policies can be defined for many domains. Such policies are not optimal but are general, can be applied efficiently (they involve no search), and often produce good results. For example, a generalized policy for the blocks world can be given as follows (see [14, 24] for better policies for this domain):

1. *put all blocks on the table, and then*
2. *move block x on block y when x should be on y, both blocks are clear, and y is well placed*

Here a block is 'well placed' when it is on the 'right' block (or table) and all blocks beneath it are well placed too. This strategy puts all blocks on the table and then moves them in order to their target positions. The strategy is completely *domain-dependent* (it only applies to the blocks-world), but is completely *instance-independent* (it applies to all the domain instances). Moreover, the length of the plans it produces

is never more than twice the length of the optimal plans [14].

Similar policies can be defined for many planning domains; e.g., [11] formulates a general policy for logistics problems. An important question is whether such policies can be inferred automatically from domain descriptions. This question has been addressed recently by Khardon [16].

3 Learning Rule-based Policies

Khardon assumes a representation of generalized policies (that he calls *action strategies*) in terms of an ordered list of existentially quantified rules. For example, some of the rules look as:

$$Obj(x), Obj(y), clear(y),\ holding(x),\\ inplace(y), G(on(x,y)) \Rightarrow Stack(x,y)$$

The left-hand side of these rules expresses the conditions that are checked in the situation, and the right-hand side expresses the action to be taken when the conditions are true for certain bindings of the variables. A *situation* is a *problem instance*; i.e., a set of object names, a state, and a goal. The conditions that are preceded by the marker G are evaluated with respect to the goal, the object predicate Obj is evaluated with respect to the set of objects, and the other conditions are evaluated with respect to the state. The rule above says to stack an object x on top of an object y, if y is clear, x is being held, y is 'in place', and x is on top of y in the goal.

Khardon refers to an ordered list of such rules as a *production rule strategy* or PRS. The action determined by a PRS in a given situation is the action in the right-hand side of the first rule whose conditions are true in the situation. If several bindings of the variables match, the first binding (in some order) is selected.

The vocabulary for the rule conditions involves the *primitive predicates* that appear in the domain description (e.g., *on*, *clear*), *support predicates* defined by the user (e.g., *above*, *in_place*), the goal marker G, and the object predicate Obj.

Khardon shows how such policies can be learned from a training set given by a set of situation-action pairs obtained by solving a number of small problem instances. The learning algorithm is variation of Rivest's algorithm for learning decision lists [22]. In this algorithm, all *candidate rules* are first enumerated, and the PRS is obtained by selecting the rule that 'best' covers the instances in the training set, and iterating this procedure over the instances that are not covered.

```
For each candidate rule
  For each example in the training set
   If rule conditions satisfied by example
    Mark that rule covers example
    Test and Mark whether covers it
        correctly or incorrectly
Initialize Rule-List to empty
While training set not empty
  choose rule that best covers the examples
  add such rule to Rule-List
  remove examples covered by rule
```

Figure 1: Rivest's Learning Algorithm: the input is the training set and the set of candidate rules; the output is an ordered list of rules

The basic idea behind the algorithm is shown in Fig. 1. The algorithm is guaranteed to find an ordered list of rules compatible with the data if such list exists [22].

For the enumeration to work, the set of candidate rules need to be finite and not too large. This is achieved by setting limits to the number of conditions and variables allowed in each rule. The size of the resulting set of rules is the main source of complexity in the learning algorithm.

Khardon reports results in two domains: blocks-world and logistics. We focus on the blocks-world. For the experiments he uses rules with two conditions and three variables. The primitive predicates are $on(x,y)$, $clear(x)$, $on\_table(x)$, and $holding(x)$ and the actions are $pickup(x)$, $putdown(x)$, $unstack(x,y)$, and $stack(x,y)$. In addition to the primitive predicates, Khardon introduces four *support* predicates that encode *background knowledge* about the domain. These include the *above* predicate and the *in_place* predicate (that expresses that a block is not misplaced). Given a training set consisting of 4800 block-world instances involving 8 blocks and their solutions, the system infers a rule-based policy that solves 79% of the 8-blocks instances, 60% of the 15-block instances, and 48% of the 20-block instances. Interestingly, the planner used as a 'teacher' (Graphplan [6]) does not solve any of the large instances.

These results are encouraging and show that the performance of a planner can be boosted by using a learning component, and that generalized policies can be learned from data. While there have been a number of proposals for learning control knowledge to guide a search engine (e.g., [27]), few approaches have aimed at learning knowledge that completely eliminates the

need to search.

The results obtained by Khardon, however, exhibit two weaknesses. First, the results are obtained using *support predicates* such as *above* and *in_place* that encode key features for solving the problem. This is domain-dependent knowledge that may be as difficult to provide as the domain-specific control knowledge that is sought. Second, the learned policies do not generalize so well: while solving 49% of the 20-block instance is better than what can be achieved by the domain-independent planner used a teacher, it is still far from what can be achieved with the simple policy for the blocks-world in which all blocks are put on the table and then stacked in order into their target positions.

In the rest of the paper we aim to show these weaknesses may be overcome by using an alternative language for representing policies.

4 Learning Concept-based Policies

Consider a control rule that says that 'if there is a clear block x that is misplaced, then move x to the table'. This rule has the form 'if an object is in a certain class, then do action a on the object' Our approach is based on the observation that rules of this form are very common in planning, and therefore, a language that makes the notion of *class* more central can provide a more compact description of generalized policies and a more convenient hypothesis space for learning them.

4.1 Concept Languages

The notion of classes of objects is central in the languages developed in AI known as *concept languages* or *description logics* [8, 9, 12]. These languages have the expressive power of subsets of standard first-order logic but with a *syntax* that is suited for representing and reasoning with classes of objects. From a logical point of view, one can think of concept languages as languages for defining complex predicates in terms of primitive ones. Predicates in concept languages are divided into two types: *unary predicates* or *concepts* that denote classes of objects, and *binary predicates* or *roles* that denote relations among objects. If we let C and C' stand for concepts, R and R' stand for roles, and C_p and R_p stand for *primitive* concepts and roles, the *complex* concepts can be defined by grammar rules of the form

$$C, C' \longrightarrow C_p \mid \top \mid \neg C \mid C \wedge C' \mid (\forall R.C) \mid R = R' \quad (1)$$

while complex roles can be defined as

$$R, R' \longrightarrow R_p \mid R_p^{-1} \mid R_p^* \mid R \circ R' \quad (2)$$

Rule (1) says that concepts can be primitive concepts, the universal concept, the complement of a concept, the intersection of two concepts, the class of individuals in the domain whose R related individuals are all member of C, or the class of individuals in the domain whose whose R related individuals and whose R' related individuals are the same. Likewise, (2) says that roles can be primitive roles, the inverse or transitive closure of primitive roles, or the composition of two roles. The semantics of these constructions is formalized by defining the interpretations (extensions) C^I and R^I of complex concepts and roles in terms of the interpretation of its constituents and the domain of discourse Δ [12, 2]. As an illustration, the semantic clauses corresponding to some of the constructions above are

$$\begin{aligned}
(\forall R.C)^I &= \{d_1 \in \Delta | \forall d_2 : \langle d_1, d_2 \rangle \in R^I \rightarrow d_2 \in C^I\} \\
(R = R')^I &= \{d_1 \in \Delta | \forall d_2 \in \Delta \rightarrow (\langle d_1, d_2 \rangle \in R^I \\
&\qquad \text{iff } \langle d_1, d_2 \rangle \in R'^I)\} \\
(R_p^{-1})^I &= \{\langle d_2, d_1 \rangle | \langle d_1, d_2 \rangle \in R_p^I\} \\
(R_p^*)^I &= \{\langle d_1, d_3 \rangle | \langle d_1, d_3 \rangle \in R_p^I \text{ or} \\
&\qquad \langle d_1, d_2 \rangle \in (R_p^*)^I \ \& \ \langle d_2, d_3 \rangle \in R_p^I\}
\end{aligned} \quad (3)$$

As an illustration, if on_s and on_g are primitive roles standing for the relation *on* in the current state s and in the goal G respectively, then $(on_s = on_g)$ is the concept that denotes the objects in the domain (blocks) that are on the same block in both s and G. Similarly, $(\forall on_g^*.\ on_s = on_g)$ is the concept that stands for the blocks that in the goal description are above blocks that are 'well placed'; i.e., all of which are on the same block in s and G. Note that equivalent descriptions in the standard syntax of predicate logic would be less compact.

4.2 Concepts for Planning

We assume that the planning domain is described in terms of a set of unary and binary predicates. For each predicate p, we create two predicates p_s and p_g: the first gets evaluated in the *state* of the given situation, the second gets evaluated in the *goal*.[1] From these *primitive* concepts and roles, the grammar above is used to generate *complex* concepts and roles. We limit the total set of concepts generated by imposing a limit on the *size* of the concepts, the size being understood as the number of connectives involved in the concept, or equivalently, the number of grammar rules needed to generate the concept. The notation $\mathcal{C}^i$ and $\mathcal{C}_i$ will be used to refer to the set of concepts of size i and size no greater than i respectively. For example, if the

[1] For predicates p that are not used in the goal description a single predicate p_s suffices.

concepts C_a and C_b have size 2 and 3, then the concept $C_a \wedge C_b$ will have size 6.

A *situation* given by a set of objects O, a state s , and a goal G provides an *interpretation* over the concept language. The domain of the interpretation Δ is given by the set of objects O, the interpretation p_s^I of the primitive predicates p_s is given by the state s, while the interpretation p_g^I of the primitive predicates p_g is given by the goal G. The interpretation C^I of all other concepts in $\mathcal{C}$ is given by the semantic rules (3). We say that a concept C is *satisfied* in a given situation I if $C^I \neq \emptyset$ and that o is an instance of C in I if $o \in C^I$.

For example, a goal description $on(a,b) \wedge on(b,c)$ defines the interpretation on_g^I of the primitive role on_g as the set containing the two pairs $\langle a,b\rangle$ and $\langle b,c\rangle$. The semantic clauses above then make the interpretation of the complex role on_g^* to be equal to the union of these two pairs and the pair $\langle a,c\rangle$.

4.3 Simple Concept-based Policies

In domains where all action predicates take a *single* argument, we define the *simple concept-based policies* as the ones expressed by an ordered list of concept-action pairs of the form:

$$C_1 : a_1, C_2 : a_2, C_3 : a_3, \ldots, C_m : a_m \qquad (4)$$

where each C_i and a_i stand for a concept and an action predicate respectively. A policy of this form says to perform an action $a_i(o)$ in a situation I when o is an instance of the concept C_i and C_i is the first concept in the list that is satisfied in I. In other words, (4) says to do action a_1 on an object o_1 if there is one such object in C_1^I, else do action a_2 on o_2 if there is one such object in C_2^I, and so on.

Such policy is *non-deterministic* in the sense that the object o_i on which the action a_i is performed is chosen non-deterministically (randomly) from the class C_i^I. So during problem solving, the policy (4) can produce different actions when the same situation is encountered more than once.[2]

The policies defined by (4) are also *incomplete* in the sense that in a given situation it may be the case that none of the concepts C_i is satisfied and therefore no action is produced. We say in that case that the policy *fails*. This problem can be avoided by setting the last concept C_n in (4) to the universal concept $\top$ that is always satisfied. However, in the algorithm below for *learning* policies of the form (4) no special provision is made for precluding policies from failing in this way, and actually they very seldom do.

We call each of the concept-action pairs $C_i : a_i$, a *rule*. Note that if there are n_p action predicates and all concepts up to size n are considered, the number of candidate rules $C_i : a_i$ is $|\mathcal{C}_n| \cdot n_p$.

4.4 Non-simple Concept-based Policies

When actions have arity greater than 1, the simple concept-based policies defined by (4) do not apply. While we don't deal with such case in this paper, we will briefly discuss it in Section 7.

4.5 Learning Simple Concept-based Policies

We focus now on the algorithm for learning simple concept-based policies. The algorithm is a slight variation of the learning algorithm depicted in Fig. 1 from [22], where rules are of the form $C_i : a_i$, and examples in the training set are of the form $I : A$, where I is a situation (set of objects, state, and goal) and A is a *set* of actions. The distinctions from Khardon's approach are thus two: the rule language based on a concept language, and the use of *sets* of actions in the examples in the training set. The examples are obtained by running a planner on a set of small random instances and collecting *all* actions that are optimal in each of the situations encountered. The planner used is a modification of the HSP planner [7].

In order to apply the learning algorithm we have to specify when a rule $C_i : a_i$ *covers* an example $I : A$ and when the rule covers the example *correctly*. Thus we say that the rule $C_i : a_i$ covers the example $I : A$ when C_i^I is not empty, and that it covers the example $I : A$ correctly when for *every* object d in C_i^I, $a_i(d)$ is in A.[3] In other words, a rule covers the example when the rule applies to the example, and covers the example correctly when the set of actions it suggests are all compatible with the actions deemed appropriate in the example.

The rules $C_i : a_i$ that *best* cover a set of examples are obtained by considering several criteria: the number N_i of examples covered incorrectly by the rule, the number N_p of examples covered correctly, and the complexity of the rule (measured by the size of C_i).

[2] Such policies can be made *deterministic* by fixing an ordering among the objects in the domain and always selecting the first such object in C_i^I. See below.

[3] This definition is suitable for the non-deterministic interpretation of concept-based policies. For the deterministic interpretation where a fixed linear ordering among the objects is assumed a priori, it is more adequate to test only the *first* object d in C_i^I rather than *every* object.

These three measures are considered lexicographically: the second measure is maximized, while the first and third measures are minimized. Ties are broken randomly.

5 Experiments

5.1 Setting

The above ideas were applied to the problem of learning generalized policies in the blocks-world. The concept language is defined by the grammar rules (1) and (2) from the set of primitive concepts (unary predicates) $clear_s$, $clear_g$ and $holding_s$,[4] and the primitive roles (binary predicates) on_s and on_g. The actions considered are $pick(b_1)$, $put\_on(b_2)$ and $put\_on\_table(b_3)$, where b_1 refers to the block to be picked, b_2 refers to the block on which the block being held is to be placed, and b_3 denotes the block that is to be placed on the table.[5]

The universe of concepts was initially restricted to the class $\mathcal{C}_7$ of concepts with size up to 7. This set, however, contains several million concepts which turned out to be too large for our current Lisp prototype. So we decided to pick a subset of $\mathcal{C}_7$ by restricting the concepts in this subset to be *conjunctive* combinations of concepts C of size up to 3. In other words, the subset of concepts considered, that we call $\mathcal{C}_{3,7}$, stands for the concepts in $\mathcal{C}_7$ of the form $C_1 \wedge C_2 \wedge \cdots \wedge C_n$ such that each 'building block' C_i is a concept in $\mathcal{C}_3$. The choice of this set of concepts is largely arbitrary; yet much smaller sets such as $\mathcal{C}_5$ with less complex concepts produced poor results, while much larger sets such as $\mathcal{C}_8$ turn out to be intractable. Later on we will discuss how the selection of a suitable hypothesis space can be automated by an iterative search procedure in which the sets $\mathcal{C}_1$, $\mathcal{C}_2$, ... are considered in sequence.

The resulting number of concepts in the hypothesis space $\mathcal{C}_{3,7}$ is 72.228 after excluding redundant concepts such as $\neg\neg C$, $C \wedge C$, etc. We also removed concepts that were not sufficiently discriminating by generating a limited number of random situations (100) and excluding all concepts whose extensions over all these situations were equivalent to the universal or null concepts. We will say more about this below.

[4]The primitive concept $holding_g$ representing the class of objects being held in the goal was not used as the predicate *holding* does not appear in the goal descriptions of the instances considered.

[5]We chose this formulation because it makes all action predicates unary. In the standard formulations, some of these actions take two arguments, but while such arguments are needed in Strips, they are not needed in other action languages; e.g., [13].

Given the 3 unary action predicates in the domain and the 72,228 concepts, the total number of candidate rules $C:a$ considered was 216.684.

The training set was generated by solving 50 random instances with 5 blocks. For each situation arising in the solution of these problems we recorded the complete *set* of actions that were found to be optimal in each situation. This produced a set of 638 examples (situation-actions pairs).

5.2 Results

In our current Lisp prototype, computing the coverage of the 216.684 rules over the set of 638 examples takes several hours. This time can probably be reduced substantially but we haven't devoted much time to optimization. From the coverage information the computation of the list of rules $C_i:a_i$ representing the policy is very fast (Fig. 1).

Fig. 2 shows one of the typical policies obtained. As it can be seen, the policy is quite compact. We have written each rule $C_i:a_i$ as $a_i(C_i)$ to improve readability. Understanding the rules, however, requires some familiarity with concept languages. For example, the concept

$$(\forall\, on_g^{-1}.holding)$$

appearing in rule A1 expresses the class of blocks B such that the block being held goes on top of B in the goal. Rule A1 thus says to put the block currently being held on a block B such that B is on the same block in the state and the goal, the block being held goes on top of B in the goal, and B is currently clear. If there is no such block B, rule A1 does not apply, and the following rules are considered in order.

The second rule A2 says to put what's being held on the table. Rule A3 is interesting and says to pick up a block B if it's clear, and its target block C is clear and *well-placed*. This last condition is captured by the subconcept $(\forall\, on_g^*.(on_g = on_s))$. Also note that in the same rule the intersection

$$(\forall\, on_g^*.\ (on_g = on_s)) \wedge (\forall\, on_g.clear_s)$$

denotes the NEXT-NEEDED-BLOCK concept; namely, the blocks whose destination block is a clear and WELL-PLACED-BLOCK. The rest of the rules admit similar readings.[6]

[6]It's important to understand the rules in the context of the other rules. E.g., rule $A1$ appears as it may place a block on top of a 'bad' tower sometimes, however, this won't happen if the rules that select the blocks to be picked, pick up the 'correct' blocks.

A1: PUT_ON $((on_g = on_s) \wedge (\forall\, on_g^{-1}.holding) \wedge clear_s)$
A2: PUT_ON_TABLE $(holding)$
A3: PICK $((\forall\, on_g^*.(on_g = on_s)) \wedge (\forall\, on_g.clear_s) \wedge clear_s)$
A4: PICK $(\neg(on_g^* = on_s^*) \wedge (\forall on_s.(\forall on_g^{-1}.clear_s)) \wedge clear_s)$
A5: PICK $(\neg(on_g = on_s) \wedge (\forall\, on_g^*.(on_s^* = on_s)) \wedge clear_s)$
A6: PICK $(\neg(on_g = on_s) \wedge (\forall\, on_s.(\forall\, on_s^{-1}.clear_s)))$

Figure 2: Policy learned from initial set of examples

5 blocks	10 blocks	15 blocks	20 blocks
96.5%	87.6%	82.8%	72.2%

Table 1: Percentage of test problems solved by policy shown in Fig. 2 for different problem sizes

The policy in Fig. 2 was tested over a large set of randomly generated block-world problems of different size: 1000 problems of 5 and 10 blocks each, and 500 problems of 15 and 20 blocks each. These instances were generated using the program BWSTATES of Slaney and Thiébaux [25]. The percentage of problems solved by this policy is shown in Table 1.

The policy solves a problem when it leads to the goal within a maximum number of steps,[7] and otherwise fails (this includes situations where no rule applies). The coverage, as it can be seen from the table, is significantly better than the coverage obtained by the rule-based policies reported by Khardon, although it is significantly worst than the 100% coverage than can be achieved by the *simple policy* of putting all blocks on the table and then stacking the blocks in order. On the other hand, the *quality* of the solutions obtained by the learned policy is better on average than the quality of the simple policy, but of course, is not as good as the quality of the *optimal* solutions that can be found. These results are shown in Table 2 that displays the number of steps taken on average by each one of these three policies.[8] As it can be seen, the quality performance of the learned policy is closer to the performance of the optimal policy than to the performance of the simple policy.

[7]This maximum number of steps was defined as four times the number of blocks. This is the maximum number of actions (*pick*'s and *put*'s) that corresponds to the policy of moving all blocks to the table and then to their target positions.

[8]Optimal solutions to these problems were obtained using the optimal block-world solver BWOPT of Slaney and Thiébaux [25].

Policy	5 blks	10 blks	15 blks	20 blks
Learned	9.98	24.32	39.77	54.94
Simple	11.57	27.74	45.19	62.68
Optimal	9.94	23.48	37.5	51.60

Table 2: Average number of steps taken by the *learned* policy over the problems that are solved, in comparison with the *simple* policy of putting all blocks in the table first, and the *optimal* policy

B1: PUT_ON $((on_g^* = on_s^*) \wedge (\forall\, on_g^{-1}.holding) \wedge clear_s)$
B2: PUT_ON_TABLE $(holding)$
B3: PICK $((\forall\, on_g^*.(on_g = on_s)) \wedge (\forall\, on_g.clear_s) \wedge clear_s)$
B4: PICK $(((on_g \circ on_g) = on_g) \wedge (\forall\, on_s.(\forall\, on_s^{-1}.clear_s)))$
B5: PICK $(((on_g \circ on_g) = on_s) \wedge (\forall\, on_s.(\forall\, on_s^{-1}.clear_s)))$
B6: PICK $(\neg(on_g^{-1} = on_s) \wedge (\forall on_s.(\forall on_g.clear_s)) \wedge clear_s)$
B7: PICK $((\forall\, on_s.clear_g) \wedge clear_s)$
B8: PICK $(\neg(on_g^{-1} = on_s) \wedge (\forall on_s.(\forall on_g^{-1}.clear_s)) \wedge clear_s)$
B9: PICK $(\forall\, on_s.(\forall\, on_s^{-1}.clear_s))$

Figure 3: Policy learned after selective extension of training set

5.3 Incremental Refinement

The learned policy shown in Fig. 2 failed on 35 of the 1000 problems in the test set with 5 blocks. For these 35 problems, we computed the optimal solutions and detected the situations arising in these problems that were not covered correctly by the policy. This resulted in the identification of 32 *new* situations-actions pairs that were added to the training set. Since the coverage of the previous 638 examples was stored, only the coverage of the new 32 examples had to be computed for each of the 216.684 candidate rules. This refinement was much faster than the original computation, leading to a policy different than the one before, shown in Fig. 3.

The results of this new policy are shown in the second row of Table 3. The row displays the percentage of problems solved by the new policy on a new set of randomly generated problems. The new policy is better than the previous policy over the small problems but is worst over the larger problems. On the smallest 5-block problems it fails on 4 of the 1000 test problems. We performed the same refinement as before by computing the optimal plans for these 4 problems, identifying 4 new situation-actions pairs that were not accounted for correctly. These 4 examples were added to the training set and the learning algorithm was rerun resulting in a third policy whose performance is shown in the third row of Fig. 3. The new policy is

better than the previous two policies and was computed very fast as only 4 new examples needed to be processed. The improvement obtained, however, is significant and has to do with the fact these few examples were not chosen randomly but were identified from the failures of the existing policy. This incremental learning process is akin to a form of *active* learning in which the training data is selected to boost performance while reducing the amount of data needed. In any case, as the second policy shows, the improvement is not monotonic.

Num. of exmpls	*Coverage of resulting policy*			
	5 blks	*10 blks*	*15 blks*	*20 blks*
638	96.5%	87.6%	82.8%	72.2%
670	99.6%	90.3%	82.0%	65.0%
674	99.7%	92.7%	84.2%	74.8%

Table 3: Percentage of problems solved by original learned policy and new policies obtained by incremental learning procedure described in text.

5.4 Deterministic vs. Non-Deterministic Policies

The results above improve the results obtained by Khardon [16] in the sense that the policies obtained produce better results while using a smaller set of examples and no background knowledge. We conjecture that this is because the concept language provides a more compact representation of generalized policies and therefore a more convenient hypothesis space for the learning algorithm to search. There is however an additional difference between Khardon's approach and ours. Khardon deals with *deterministic* policies where every situation produces a unique action and we deal with *non-deterministic* policies. We performed a quick test to evaluate how relevant this difference is by computing the coverage and quality of the learned policy shown in Fig. 2 when applied *deterministically*. For that, we fixed an ordering among the objects in the domain and always selected to act upon the first object in this ordering that satisfied the first concept in the policy list. The *coverage* of the resulting execution of the policy got indeed reduced while the average *quality* of the resulting plans improved slightly. The largest difference in coverage occurred in the largest problems. For the 20-block problems, the deterministic execution of the policy produced solutions for 59% of the problems, down from the 72.2% achieved by the non-deterministic execution. This is because the deterministic execution is more likely to get trapped into loops. By possibly choosing different actions when the same situation is encountered more than once, the non-deterministic execution breaks such loops. This also explains why the deterministic execution produces slightly shorter plans on average on the problems that it solves. The coverage of deterministic policies, however, can be improved by taking the ordering of the objects into account not only during the testing phase but also during the learning phase (when computing the coverage of each candidate rule). We are currently performing such experiments.

6 Related Work

This work presents an approach for learning generalized policies from data that is a variation of the approach taken by Khardon in [16]. Khardon represents generalized policies (action strategies) in a rule language and learns the policies in a supervised manner using a variation of Rivest's decision list learning algorithm [22] and additional background knowledge. We use the same learning algorithm but represent policies in terms of a concept language. The motivation for this is that planning involves the selection of the 'right' actions on the 'right' classes of objects, and concept languages provide a convenient, compact syntax for defining the useful classes of objects and learning them. Through a number of experiments in the blocks-world we have shown that the same learning algorithm produces better policies with less data and no background knowledge. We expect that similar results can be obtained in other richly structured domains even though there are a number of obstacles that must be overcome (see Sect. 7).

The approach described in this paper (as well as Khardon's) applies not only to classical, deterministic planning domains but also to *stochastic* planning domains where actions can be probabilistic.[9] In that case, the examples in the training set must be obtained by a dynamic programming procedure rather than by a classical planner. No other changes are required and the same procedures can yield suitable policies over such domains. In this sense, it's worth comparing this approach with *reinforcement learning* methods that are also concerned with learning policies for classical and stochastic planning problems [26]. One difference is that our approach is *supervised* as it relies on the existence of set of small *solved* instances while reinforcement learning (RL) is *unsupervised* (it doesn't need

[9] A (fully observable) planning problem with probabilistic action is a Markov Decision Process (MDP) whose solution can be cast as a policy that maps states into actions [5, 23].

a teacher). A second difference is that in reinforcement learning the representation of the policy takes the form of a *value function*. While value functions can generalize across different states and even across different goal descriptions, they cannot cope easily with changes in the size of the state space as when we move from a problem with 5 blocks to a problem with 20 blocks. Thus, value functions do not appear suitable for learning *generalized* policies.

The concept-based representation of generalized policies has the flavor of the functional-indexical representations of policies advocated by Agree and Chapman [1]. However, in Agree's and Chapman's approach, useful concepts like the NEXT-NEEDED-BLOCK are defined by the programmer and their denotation is assumed to be provided by sensors. In the approach described here, the concepts are learned from the examples and their denotation is given by their logical structure. The two ideas, however, are related; a concept-based policy can be executed faster when the denotation of the top concepts needs not be inferred from the denotation of the basic predicates but is provided directly by sensors. Thus the top concepts appearing in the policy express the aspects of the world that are actually worth sensing. Moreover, in the concept language as in functional-indexical representations, objects are not referred by their particular names (e.g., block A, B, etc) but by the role they play in going from the current state to the goal.

The use of learning algorithm on top of rich, logical representations, relates this work to the work in *Inductive Logic Programming*. For example, FOIL is a system that learns first-order rules from data in a supervised manner [21]. Like Rivest's learning algorithm, FOIL selects the 'best' rules one by one, eliminating the examples that are covered, and iterating this procedure on the examples that are left. The main difference is in the way the 'best' rules are selected. In Rivest's algorithm, this is done by an *exhaustive* evaluation of *all* rules. This provides a completeness guarantee (if there are decision lists consistent with the data, the algorithm will find one such list) but does not scale up to very large rule sets. The greedy approach, on the other hand, does not provide guarantees but can be used over much larger rule sets. We have followed the first approach.

Finally, it's worth mentioning the work on computing generalized policies by evolutionary methods. For example, both Koza in [17] and Baum in [4], develop evolutionary algorithms for obtaining generalized policies for versions of the blocks world. In Koza's approach, the policies are represented by Lisp programs, while in Baum's approach they are represented by a collection of rules in a syntax adapted to the application. While the results obtained in both cases are good, a comparison is not easy as both approaches appear to rely on domain-specific background knowledge in the form of useful domain features ('the-next-needed-block' [17]; 'top-of-the-stack' [4]) or domain-specific state representations. Like RL methods, these evolutionary methods are non-supervised, but unlike RL methods, they can compute generalized policies over all problem instances in the domain.

7 Discussion

Building on the work of Khardon [16], we have presented a representation for generalized policies based on a concept language and have shown through a number of experiments in the blocks-world that the new representation appears to produce better policies with less training data and no background knowledge.

The challenge is to show that these ideas can be applied successfully to other structured planning domains. The main obstacle is the combinatorics of the learning algorithm that requires computing the denotation of each one of the concepts in $\mathcal{C}_n$ in each one of the situations in the training set. The number $|\mathcal{C}_n|$ of concepts of size up to n grows exponentially with n. Currently, we consider very small values of n and exclude a number of redundant concepts such as $\neg\neg C$, $C \wedge C$, etc. However this pruning is not enough in general. We could use a more complete subsumption test over concepts but it's not clear whether such tests are cost-effective. An alternative approach that we plan to explore in the future is to generate the sets $\mathcal{C}_n$ incrementally, for $n = 0, 1, \ldots$, pruning all concepts that 'appear' equivalent to simpler concepts. This can be done quickly over a set of random examples: two concepts 'appear' equivalent when they have the same extensions over each one of the examples. While such pruning is not sound, it may allow us to deal with richer domains with more predicates with a small penalty in the quality of the policies obtained. An alternative that seems less appealing is to move away from Rivest's algorithm, replacing the exhaustive search and evaluation of *all* possible rules by an incomplete search.

Another limitation of the approach presented in this paper is the arbitrary choice of the set $\mathcal{C}_n$ of concepts considered. However, the incremental pruning approach can provide a solution to this problem: rather than chosing the value of n a priori, we can do an iterative search procedure, evaluating the policies resulting from the set of concepts $\mathcal{C}_1$, $\mathcal{C}_2$, ... in sequence, until

a good policy is found.

A more serious limitation is the restriction to *unary* actions. One option there is to break actions of higher arity into several actions of lower arity and suitable fluents and preconditions. This transformation can be done automatically in a domain-dependent manner but it may not be effective in general. A second option is to extend the rules $C_i : a_i$ in the simple policies (4) by rules of the form $C_i^1, C_i^2, \ldots, C_i^m : a_i$, where m is the arity of the action predicate a_i. The tuple of arguments upon which the action a_i is done is then selected from the corresponding tuple of concepts. This idea is simple but does not exploit the fact that the action arguments are usually functionally related (e.g., move a clear block to *its* target location). A more promising approach is to allow the concept C_i^k that stands for the k-th argument of the action to refer to the previous arguments C_i^j, $j < k$. Thus, if the number m is a reference to the m-th action argument, an action like 'moving a clear block to its target position' would be expressible as $move(clear_s, (\forall on_g^{-1}.\ 1))$.

A final issue that is worth mentioning is that in some domains more powerful concept languages may be needed; e.g., languages involving number restrictions or min-max operators. We hope to address this and the other limitations discussed in future work.

Acknowledgments

Part of this work was done while H. Geffner was visiting IRIT, Toulouse, and Linköping University. He thanks J. Lang, D. Dubois, H. Prade, and H. Farreny in Toulouse, and E. Sandewall and P. Doherty in Linköping for their hospitality and for making these visits possible. We also thank J. Slaney and S. Thiébaux for making the programs BWSTATES and BWOPT available. The work has been partially supported by grant S1-96001365 from Conicit, Venezuela and the Wallenberg Foundation, Sweden.

References

[1] P. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *Proceedings AAAI-87*, 1987.

[2] F. Baader and U. Sattler. Number restrictions on complex roles in description logics. In *Proceedings KR'96*. Morgan Kaufmann, 1996.

[3] F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning, 1998. Submitted.

[4] Eric Baum. Toward a model of mind as a laissez-faire economy of idiots. In *Proceedings Int. Conf. on Machine Learning*, 1996.

[5] D. Bertsekas. *Dynamic Programming and Optimal Control, Vols 1 and 2.* Athena Scientific, 1995.

[6] A. Blum and M. Furst. Fast planning through planning graph analysis. In *Proceedings of IJCAI-95*, Montreal, Canada, 1995.

[7] B. Bonet and H. Geffner. Planning as heuristic search: New results. In *Proceedings of ECP-99*. Springer, 1999.

[8] R. Brachman and H. Levesque. The tractability of subsumption in frame based description languages. In *Proceedings AAAI-84*, 1984.

[9] R. Brachman and J. Schmolze. An overview of the KL-ONE knowledge representation systems. *Cognitive Science*, 9(2), 1985.

[10] T. Bylander. The computational complexity of STRIPS planning. *Artificial Intelligence*, 69:165–204, 1994.

[11] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings KR'91*. Morgan Kaufmann, 1991.

[12] H. Geffner. Functional strips: a more general language for planning and problem solving. Presented at the Logic-based AI Workshop, Washington D.C., June 1999.

[13] N. Gupta and D. Nau. Complexity results for blocks-world planning. In *Proceedings AAAI-91*, pages 629–633, 1991.

[14] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of AAAI-96*, pages 1194–1201, 1996.

[15] R. Khardon. Learning action strategies for planning domains. Technical Report TR-09-97, Harvard, 1997. To appear in AI Journal.

[16] J. Koza. *Genetic Programming I.* MIT Press, 1992.

[17] D. McDermott. PDDL – the planning domain definition language. Available at `http://ftp.cs.yale.edu/pub/mcdermott`, 1998.

[18] D. Nau, Y. Cao, A. Lotem, and H. Munoz-Avila. Shop: Simple hierarchical ordered planner. In *Proceedings IJCAI-99*, 1999.

[19] B. Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34(3), 1988.

[20] N. Nilsson. *Principles of Artificial Intelligence.* Tioga, 1980.

[21] J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[22] R. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

[23] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1994.

[24] J. Slaney and S. Thiébaux. Linear time near-optimal planning in the blocks world. In *Proceedings of AAAI-96*, pages 1208–1214, 1996.

[25] J. Slaney and S. Thiébaux. Software for the block-world: BWSTATES and BWOPT. At `http://arp.anu.edu.au/ jks/bw.html`, 1999.

[26] R. Sutton and A. Barto. *Introduction to Reinforcement Learning.* MIT Press, 1998.

[27] M. Veloso, J. Carbonell, A. Perez, D. Borrajo, E. Find, and J. Blythe. Integrating learning and planning: The prodigy architecture. *J. of Experimental and Theoretical AI*, 1994.

Planning with sensing, concurrency, and exogenous events: logical framework and implementation

Luca Iocchi, Daniele Nardi, Riccardo Rosati
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
<iocchi,nardi,rosati>@dis.uniroma1.it

Abstract

The focus of current research in cognitive robotics is both on the realization of systems based on known formal settings and on the extension of previous formal approaches to account for features that play a significant role for autonomous robots, but have not yet received an adequate treatment. In this paper we adopt a formal framework derived from Propositional Dynamic Logics by exploiting their formal correspondence with Description Logics, and present an extension of such a framework obtained by introducing both concurrency on primitive actions and autoepistemic operators for explicitly representing the robot's epistemic state. We show that the resulting formal setting allows for the representation of actions with context-dependent effects, sensing actions, and concurrent actions, and address both the presence of exogenous events and the characterization of the notion of executable plan in such a complex setting. Moreover, we present an implementation of this framework in a system which is capable of generating plans that are actually executed on mobile robots, and illustrate the experimentation of such a system in the design and implementation of soccer players for the 1999 Robocup competition.

1 INTRODUCTION

The focus of research in Cognitive Robotics is in devising actual implementations of reasoning about actions and action planning on top of mobile robots. To this purpose, several approaches have been proposed, based on various extensions of the Situation Calculus [Lesperance *et al.*,1998], Propositional Dynamic Logics (PDLs) [De Giacomo *et al.*,1996; 1997b], the Event Calculus [Shanahan,1997], and $\mathcal{A}$-languages and logic programming [Gelfond and Lifschitz,1993]. In pursuing this goal, two kinds of difficulties arise. On the one hand, the practical realization of formalisms on top of mobile robots requires the designer to put restrictions either on the expressiveness of the language or on the role played by automated reasoning. On the other hand, several features that have often been neglected by previous work on representation of dynamic systems and reasoning about actions are now receiving attention, because they are needed in practice: e.g., reasoning about the concurrent execution of actions [Lin and Shoham,1992; Baral and Gelfond,1993], sensing actions [Levesque,1996], and exogenous events [De Giacomo *et al.*,1997a].

In this paper we follow the approach to the representation of dynamic systems based on PDLs [Rosenschein,1981] and its extension based on the correspondence with Description Logics [De Giacomo *et al.*,1996; 1997b]. In fact, we extend such an approach in order to address action planning in the presence of sensing actions, concurrent actions and exogenous events, in the realm of Cognitive Robotics, namely with the goal of implementing it on top of a mobile robot.

An analysis of the state of the art in Cognitive Robotics shows that, while several semantic questions concerning the representation of complex dynamic domains have been addressed (see e.g. [Lin and Shoham,1992; Baral and Gelfond,1993; Reiter,1996; Pinto,1998] for the issue of representing concurrent actions), the problem of generating plans which can be actually executed by a robotic architecture is generally not considered. We thus propose a rich formal setting, where one can deal with sensing actions and concurrent execution of actions, and express static constraints and frame axioms with different flavors, still

retaining the ability of generating plans which can be actually executed by mobile robots. Moreover, we have experimented our approach in a real Cognitive Robotics application, namely the RoboCup99 F-2000 competition for soccer-playing robots, within the Azzurra Robot Team [Nardi *et al.*,1999].

The paper is organized as follows. In the next section we summarize the main features of the epistemic approach to representing dynamic systems. Then we discuss the representational features of our framework and, subsequently, reasoning. Finally, we report on the experimentation of our approach in the RoboCup competition.

2 AUTOEPISTEMIC APPROACH TO ACTION REPRESENTATION

Generally speaking, our work on the definition of the logical framework is based on the idea of using a nonmonotonic formalism in order to logically reconstruct a number of expressive features for action representation. More specifically, we have analyzed the modal nonmonotonic logic $\mathcal{ALCK}_{\mathcal{NF}}$, and experimented its adequacy in the formalization of dynamic systems. Notably, it can be shown that such a formalism is able to logically capture and extend several formal frameworks for action representation (e.g., STRIPS and the language $\mathcal{A}$).

We start by pointing out the main capabilities that the logical framework has to provide, in order to fully support a suitable representation for the Cognitive Robotics applications under our examination.

Epistemic abilities First of all, we require a formalism in which it is possible to *explicitly* represent the robot's epistemic state. The reason for such a requirement is twofold:

1. an explicit representation of the robot's epistemic state allows for an easy characterization of effective plans, i.e. plans which can be actually executed by the robot (see Section 4);
2. as shown e.g. in [Scherl and Levesque,1993], an explicit representation of the robot's epistemic state allows for a natural formalization of sensing (or knowledge-producing) actions, namely actions which allow the robot to know the value of a property in the current state of the world. The peculiarity of such actions lies in the fact that their execution only affects the robot's knowledge about the world, without changing the state of the external world.

Indeed, several epistemic approaches to reasoning about actions have been proposed in the literature, as well as the use of epistemic modalities for dealing with the representation of sensing actions (see e.g. [Scherl and Levesque,1993; Lakemeyer and Levesque,1998; Lobo *et al.*,1997]). However, such proposals do not address the issue of characterizing effective plans by exploiting the use of epistemic modal formalisms.

Concurrency We require the formalism to allow for reasoning about the concurrent execution of primitive actions, without introducing the notion of time to specify the start and the end of an action such as in [Pinto,1998; Reiter,1996] or adopting interleaving semantics [De Giacomo *et al.*,1997a]. We remark that we do not want to treat in an ad-hoc way each possible concurrent execution of actions, i.e., we do not define specific axioms for each possible combination of primitive actions: instead, we want to treat concurrency in a systematic way, deriving the possibility of concurrently executing two or more primitive actions directly from the specification of such actions.

Our formalization of concurrent actions is based on the following simple principle (for ease of exposition, below we illustrate the case of $n = 2$ concurrent actions: the case of an arbitrary n is obtained by a straightforward generalization).

Definition 1 *Two actions a_1, a_2 can be concurrently executed in a state s if and only if the following conditions hold: (i) both a_1 and a_2 can be executed in s; (ii) the effects of a_1 and a_2 are mutually consistent.*

Thus, as mentioned above, the possibility of concurrently executing two primitive actions only depends on the specification (in terms of preconditions and postconditions) of such primitive actions. In particular, the first condition imposes that a_1 and a_2 can be executed in s only if they can be individually executed in s, while the second condition prevents the concurrent execution of actions whose effects are not jointly consistent. We stress the fact that, while the first condition is independent of the current state s, the second condition actually depends on s, if, in the formalization, we allow context-dependent effects of actions (see next section) and/or inertia laws.

Other approaches have proposed the formalization of concurrency of primitive actions [Baral and Gelfond,1993; Giordano *et al.*,1998; Lin and Shoham,1992]. A novel feature of our approach is the possibility of modeling the concurrent execution of primitive actions and sensing actions.

Persistence and exogenous events We also require

the formalism to allow for the specification of suitable mechanisms (frame axioms) for expressing persistence of properties. In particular, we want to be able to formalize both *deterministic frame axioms*, stating that a certain property is guaranteed to persists (in the robot's epistemic state) after the execution of an action, and *default frame axioms*, stating that a property persists after the execution of an action if it is consistent with the effects of the action.

In this way, the formalism is able to represent both monotonic and nonmonotonic solutions to the frame problem, based on the automatic generation of frame axioms starting from the specification of the dynamic system (e.g. see [Reiter,1991]). However, we recall that such automatic solutions are effective only in simple dynamic settings satisfying severe restrictions on both the effects of actions and the static relationships between properties. Furthermore, we are interested in dynamic environments in which some properties may change due to events (usually called *exogenous events*) that cannot be predicted by the robot. As we shall see in next section, this requires the possibility of formalizing non-inertial properties as well.

State and causal constraints Finally, we require the possibility of expressing both *state constraints* [Lin and Reiter,1994], that is, ordinary first-order sentences expressing static relationships between dynamic properties, and *epistemic constraints*, which can be seen as relationships which must be satisfied in each robot's epistemic state. In the following we will show that some forms of constraints that take into account *causal* dependencies between properties [Lin,1995; Mc Cain and Turner,1995; Thielscher,1997] can be considered as special forms of epistemic constraints, and will use such form of rules for specifying special kinds of frame axioms (epistemic frame axioms). Both ordinary state constraints and casual constraints enforce *ramifications*, e.g. indirect effects of actions.

Summarizing, to our knowledge none of the existing formal approaches to reasoning about actions in Cognitive Robotics is able to address *all* the above requirements for action representation.

The logic $\mathcal{ALCK}_{\mathcal{NF}}$ The formalism we employ for representing actions is basically the one presented in [De Giacomo *et al.*,1997b], extended with role conjunction in order to formalize the concurrent execution of primitive actions. Such a formalism is based on propositional dynamic logics (PDLs), and makes use of the tight correspondence between PDLs and description logics (DLs) [De Giacomo and Lenzerini,1994] that allows for considering PDLs and DLs as notational variants of each other. We use the notation of DLs, focusing on the well-known DL $\mathcal{ALC}$, corresponding to the standard PDL with atomic programs only. In addition, we use two nonmonotonic modal operators: a *minimal knowledge operator* $\mathbf{K}$ and a *default assumption operator* $\mathbf{A}$. These are interpreted according to the nonmonotonic modal logic *MKNF* [Lifschitz,1994], and give rise to the so-called autoepistemic description logic $\mathcal{ALCK}_{\mathcal{NF}}$ [Donini *et al.*,1997], which thus corresponds to a fragment of first-order *MKNF*.

Due to space limitations, we only briefly introduce $\mathcal{ALCK}_{\mathcal{NF}}$, and refer to [Donini *et al.*,1997] for a detailed introduction to such a formalism. The logic $\mathcal{ALCK}_{\mathcal{NF}}$ allows for representing a domain of interest in terms of *concepts* and *roles*. Concepts model classes of *individuals*, while roles model relationships between classes. Starting with atomic concepts and atomic roles, which are concepts and roles described simply by a name, complex concepts and roles can be built by means of the following abstract syntax:

$$\begin{array}{rcl} C & ::= & \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \neg C \mid \\ & & \exists R.C \mid \forall R.C \mid \mathbf{K}C \mid \mathbf{A}C \\ R & ::= & P \mid P_1 \sqcap \ldots \sqcap P_n \mid \mathbf{K}R \mid \mathbf{A}R \end{array}$$

where A denotes an atomic concept, C (possibly with a subscript) denotes a concept, P (possibly with a subscript) denotes an atomic role, and R denotes a role. An *$\mathcal{ALCK}_{\mathcal{NF}}$-knowledge base* is the union of a set of inclusion assertions (called the TBox) and a set of instance assertions (called the ABox). Inclusion assertions have the form $C \sqsubseteq D$, where C, D are concept expressions. Instance assertions have the form $C(a)$ or $R(a, b)$, where C is a concept, R is a role, and a, b are *names* of individuals. We assume that different names denote different individuals.

As for the semantics (which is formally defined in the appendix), an interpretation structure in $\mathcal{ALCK}_{\mathcal{NF}}$ corresponds to a possible-world (Kripke-style) structure, in which each world corresponds to an interpretation of standard DLs. The interpretation structures of DLs are essentially graphs labeled both on nodes and edges. Nodes correspond to individuals: each node is labeled by concepts that denote the properties of the individual. Edges (called links in DL) are labeled by roles. To our purposes, it is important to point out how such interpretation structures can be put in correspondence with the states of a dynamic system (due to space limitations, we refer [De Giacomo *et al.*,1997b] for a more detailed description of this aspect).

Due to the form of the assertions used in the formalization of the dynamic system, $\mathcal{ALCK}_{\mathcal{NF}}$ models of a KB Σ can be interpreted as graphs. In particular, individuals represent states of the system and are labeled by

concepts representing the properties (or fluents) that hold in that state; edges between individuals represent transitions between system states, and are labeled by roles representing the actions that cause the state transition. More specifically, each node (individual) denotes a different *epistemic* state of the robot: an edge labeled by an action R connects two such states (individuals) s, s' if the execution of R, when the robot's epistemic state is s, changes its epistemic state to s'.

As shown in [De Giacomo *et al.*,1997b], an epistemic inclusion assertion $\mathbf{K}C \sqsubseteq D$ can be naturally interpreted in $\mathcal{ALCK}_{\mathcal{NF}}$ in terms of a *rule*, i.e. a forward reasoning mechanism. For this reason, there is a strict semantic analogy between the sentence $\mathbf{K}C \sqsubseteq D$ and a *causal rule* [Mc Cain and Turner,1995] of the form $C \Rightarrow D$, and, in general, between epistemic constraints and causal relationships. Moreover, the default assumption operator $\mathbf{A}$ allows for expressing justifications of default rules, and the combined usage of $\mathbf{K}$ and $\mathbf{A}$ allows for formalizing defaults in terms of modal formulas [Lifschitz,1994]. In particular, the default rule $\frac{C(x):D(x)}{E(x)}$ (which can be interpreted as a special inference rule stating that, for each individual x, if $C(x)$ holds and it is consistent to assume $D(x)$, then $E(x)$ must be derived) can be represented in $\mathcal{ALCK}_{\mathcal{NF}}$ by means of the inclusion assertion $\mathbf{K}C \sqsubseteq \mathbf{A}\neg D \sqcup \mathbf{K}E$.

We can thus summarize the main features of $\mathcal{ALCK}_{\mathcal{NF}}$ for action representation:
(i) it allows for representing actions (through roles), states (through individuals), and state properties (through concepts);
(ii) it allows for explicitly representing the robot's knowledge, by means of two distinct autoepistemic modalities in the language;
(iii) it allows for a representation of sensing actions, through the use of the minimal knowledge operator;
(iv) it allows for the representation of concurrent actions, by allowing conjunction of binary roles;
(v) it allows for the representation of both default persistence of properties, through the use of the autoepistemic modalities, and non-inertial properties;
(vi) it allows for the representation of both ordinary state constraints and casual constraints.

3 REPRESENTING ACTIONS

In this section we present our formalization of actions in the $\mathcal{ALCK}_{\mathcal{NF}}$ framework. Specifically, we represent a dynamic system in terms of an $\mathcal{ALCK}_{\mathcal{NF}}$ knowledge base $\Sigma = \Gamma_S \cup \Gamma_I \cup \Gamma_P \cup \Gamma_E \cup \Gamma_{DFR} \cup \Gamma_{EFR}$, where each Γ_x is defined below. In the following, we assume to deal with a set of primitive actions, partitioned into a set of *ordinary* actions, which are assumed to be deterministic actions with (possibly) context-dependent effects, and *sensing* actions, which are assumed to be actions able to sense boolean properties of the environment. We represent primitive actions in $\mathcal{ALCK}_{\mathcal{NF}}$ through a set of atomic role symbols, which we call *action-roles*.

State constraints

We call Γ_S the set of state constraints (also known as domain constraints) of our formalization. State constraints are used for representing background knowledge, which is invariant with respect to the execution of actions. We formalize state constraints as general $\mathcal{ALC}$ inclusion assertions, not involving action-roles, although in general they can involve other roles used for structuring concept (i.e. property) descriptions and form complex taxonomies of properties that are typical of DLs.

Initial state We denote as Γ_I the specification of the initial state in our formalization. Such a specification is given in terms of an instance assertion of the form $C(\mathit{init})$, where C is an $\mathcal{ALC}$ concept, and *init* is the constant denoting the initial state. This axiom can be read as: C holds in the state *init* in every possible interpretation.

Precondition axioms for primitive actions We denote as Γ_P the set of precondition axioms which specify sufficient conditions for the execution of primitive actions in a state. Precondition axioms are expressed through autoepistemic sentences of the form:

$$\mathbf{K}C \sqsubseteq \exists \mathbf{K}R.\top \qquad (1)$$

where C is an $\mathcal{ALC}$ concept and R is a primitive action (either an ordinary or a sensing action). This axiom can be read as: if C holds in the current state s, then there exists a state s' which is the R-successor of s.

We remark the fact that the use of such form of precondition axioms assumes the possibility of identifying *sufficient* conditions for the execution of an action. Hence, this formalization is not able to deal with implicit qualifications: namely, if the execution of R in a state in which C holds gives rise to an inconsistent successor state (which can be caused by the interaction of action effects and state and epistemic constraints), then there is no interpretation structure satisfying the $\mathcal{ALCK}_{\mathcal{NF}}$ knowledge base, since the above precondition axiom enforces the existence of a successor state. Therefore, we conclude that the specification of the dynamic system is inconsistent.

Precondition axioms for concurrent actions In

order to formalize concurrent actions according to Definition 1, we introduce in our formalization of the dynamic system an autoepistemic axiom schema which states the preconditions under which two or more actions can be concurrently executed. In particular, we enforce the following axiom schema:

$$\exists \mathbf{K} R_1.\top \sqcap \exists \mathbf{K} R_2.\top \sqcap \neg \mathbf{A}(\forall R_1 \sqcap R_2.\bot) \sqsubseteq \exists \mathbf{K}(R_1 \sqcap R_2).\top \quad (2)$$

which encodes the preconditions for the concurrent execution of primitive actions R_1 and R_2 according to Definition 1. In fact, for each pair of actions R_1, R_2, and for each state s, R_1 and R_2 are concurrently executed in s if and only if both R_1 and R_2 are executed in s, and it is consistent to assume $\neg \forall R_1 \sqcap R_2.\bot$, i.e. the effects of R_1 and R_2 in s are mutually consistent. In the following, we implicitly assume that each $\mathcal{ALCK}_{\mathcal{NF}}$ knowledge base representing a dynamic system with primitive actions $R_1, \ldots, R_n$ contains all instances of the axiom schema $\exists \mathbf{K} R_{i_1}.\top \sqcap \ldots \sqcap \exists \mathbf{K} R_{i_k}.\top \sqcap \neg \mathbf{A}(\forall R_{i_1} \sqcap \ldots \sqcap R_{i_k}.\bot) \sqsubseteq \exists \mathbf{K}(R_{i_1} \sqcap \ldots \sqcap R_{i_k}).\top$, for $1 \leq k \leq n$, $1 \leq i_1 \leq \ldots \leq i_k \leq n$.

Notice that the above axiom schema is the only one needed in order to model concurrent actions in our formalization. In fact, due to the semantics of role conjunction, $R_1 \sqcap R_2(s, s')$ implies both $R_1(s, s')$ and $R_2(s, s')$, hence (due to the form of effect axioms) the effect of the concurrent execution of actions is modeled by the effect axioms relative to the primitive actions R_1 and R_2. For the same reason, persistence of properties during the concurrent execution of R_1 and R_2 is modeled by the frame axioms for R_1 and R_2.

Effect axioms Effect axioms specify the effects of executing an action in a given state. In order to specify the effects of actions, we distinguish between ordinary actions and sensing actions. First, ordinary effect axioms specify the effects of the execution of an ordinary action in a state satisfying given conditions. In our framework, we have that, if in the current state the property C holds, then, after the execution of the action R, the property D holds, in the following way:

$$\mathbf{K} C \sqsubseteq \forall R.D$$

Notice that, since we allow multiple effect axioms for the same action, we are able to formalize *context-dependent* actions, that is, the effect of an action may be different in two different states. Moreover, effect axioms for sensing axioms are of the form

$$\top \sqsubseteq \mathbf{K}(\forall R_S.D) \sqcup \mathbf{K}(\forall R_S.\neg D),$$

and specify the fact that, after the execution of the sensing action R_S, the robot knows whether the property D is true or false.

We call Γ_E the set of effect axioms in the formalization of the dynamic system.

Frame axioms Using the autoepistemic operators, it is possible to enforce different forms of inertia laws. In particular, we are able to specify *default frame axioms* i.e. default persistence rules which state that, if in the current state the property C holds, then, after the execution of the action R, the property C holds, if it is consistent with the effects of R. Due to the above mentioned possibility of expressing defaults in $\mathcal{ALCK}_{\mathcal{NF}}$, such a rule can be represented in our framework by the inclusion axiom

$$\mathbf{K} C \sqsubseteq \forall \mathbf{K} R.\mathbf{A} \neg C \sqcup \mathbf{K} C$$

and call Γ_{DFR} the set of default frame axioms in our specification. We also use *epistemic frame axioms* in our specification, which are able to express "causal" persistence rules. Such an axiom has the form

$$\mathbf{K} C \sqsubseteq \forall \mathbf{K} R.\neg \mathbf{K} D \sqcup \mathbf{K} C$$

and expresses the fact that property C is propagated only if property D holds in the successor state. We call Γ_{EFR} the set of epistemic frame axioms in Σ.

By suitably instantiating the above kinds of axioms in our system specification, we are able to formalize both inertial and non-inertial properties, and both inertial and non-inertial actions, thus addressing both the frame problem and the presence of exogenous events in our framework.

Notice that this kind of formalization correctly addresses the concurrent execution of a sensing action and an ordinary action, which is illustrated by the following example.

Example 2 Suppose the ordinary action R has the effect $\neg C$, while the sensing action S allows for knowing the truth value of the property D, and suppose both actions can be executed in those states in which the property C holds. Hence, Σ includes the precondition axioms $\mathbf{K} C \sqsubseteq \exists \mathbf{K} R.\top$ and $\mathbf{K} C \sqsubseteq \exists \mathbf{K} S.\top$, and the effect axioms $\top \sqsubseteq \forall R.\neg C$ and $\top \sqsubseteq (\mathbf{K} \forall S.D) \sqcup (\mathbf{K} \forall S.\neg D)$. Now suppose that all sensing actions are inertial: This can be formalized by adding to the specification a set of frame axioms of the form

$$\mathbf{K} C \sqsubseteq \forall \mathbf{K} S.\mathbf{A} \neg C \sqcup \mathbf{K} C \quad (3)$$

for each property C and for each sensing action S.

Suppose now that, in the current state (s) C is known by the robot: hence, both R (with successor state s_1)

and S (with successor state s_2) can be executed. Then, by Definition 1, the concurrent execution of R and S can be done in the current state (with successor state s_3). The effect of $R \sqcap S$ in s_3 is $\neg C \sqcap D$ (or $\neg C \sqcap \neg D$). Given our action formalization, the frame axiom (3) affects both s_2 and s_3, since both $\mathbf{K}S(s, s_2)$ and $\mathbf{K}S(s, s_3)$ hold. However, such an axiom causes the propagation of the property C only in the state s_2 and not in s_3, since in s_3 it is not consistent to assume that C holds, thus obtaining the desired behavior. □

Therefore, the $\mathcal{ALCK}_{\mathcal{NF}}$-based framework for action formalization matches the representational requirements illustrated in Section 2. Hence, it constitutes the first formal framework that allows for representing *all* the required features of a dynamic system, and provides for reasoning methods applicable to plan generation, as shown in the next section.

4 PLANNING IN $\mathcal{ALCK}_{\mathcal{NF}}$

In this section we study planning in the $\mathcal{ALCK}_{\mathcal{NF}}$ framework and present a reasoning method for generating both conditional and concurrent plans.

The planning problem The classical formulation of the notion of plan existence corresponds to the following: given the specification of the dynamic system and of the initial state, and a goal (property) G, there exists a plan for G if and only if there is a sequence of actions $\mathcal{S}$ such that the execution of $\mathcal{S}$ leads to a state s in which the goal G holds.

In order to have an effective notion of plan in our framework, we have to modify the above notion in many respects.

1. We require that the robot is aware of the fact that the goal is reached, i.e. we want the robot to know that G holds after the execution of $\mathcal{S}$.
2. We require that the plan can be effectively executed by the robot: this implies that, for each i, if s_{i+1} is the state resulting from the execution of the i-th action of the sequence $\mathcal{S}$ in state s_i, then the robot knows that the action can be executed in s_i (i.e. it knows that the precondition of the action holds in state s_i).
3. The execution of a sensing action in a given state may have in general multiple outcomes. Hence, the presence of sensing actions requires to reformulate the above notion of planning problem. In particular, following [Levesque,1996], in presence of boolean sensing actions we define a plan as a *conditional plan*, that is a program composed of action statements and if-then-else statements, such that each if-then-else statement occurs right after a sensing action statement, and is conditioned by the truth value of the sensed property. A plan for G is a conditional plan $\mathcal{S}_G$ such that the execution of $\mathcal{S}_G$ leads to a state in which G is known to hold for each possible outcome of the sensing actions in $\mathcal{S}_G$.
4. The possibility of having concurrent executions of actions further extends the above notion of conditional plan to programs containing statements denoting the concurrent execution of primitive actions.

It can be shown that the above notion of plan existence can be reduced in the $\mathcal{ALCK}_{\mathcal{NF}}$ framework to a reasoning problem:

$$\Sigma \models C_G(init) \tag{4}$$

where C_G is *any* concept belonging to the set $\mathcal{P}_G$ defined inductively as follows: (i) $\mathbf{K}G \in \mathcal{P}_C$; (ii) if $C_1, \ldots, C_m \in \mathcal{P}_C$, then $\exists \mathbf{K}(R_{M_1} \parallel \ldots \parallel R_{M_k} \parallel R_{S_1} \parallel \ldots \parallel R_{S_l}).(\mathbf{K}S_1 \sqcap \ldots \sqcap \mathbf{K}S_l \sqcap C_1) \sqcup \ldots \sqcup (\mathbf{K}\neg S_1 \sqcap \ldots \sqcap \mathbf{K}\neg S_l \sqcap C_m) \in \mathcal{P}_C$ for each $0 \leq k, l \leq n$, where $m = 2^{k+l}$, each R_{M_i} is an ordinary action and each R_{S_i} is a sensing action for the property S_i.

Remark. The above characterization intentionally does not take into account *implicit* (or indirect) forms of nondeterminism in the execution of an action in a given state. In fact, due to the presence of both state (and epistemic) constraints and default frame axioms, even primitive, ordinary actions may give rise in general to multiple possible epistemic states. Such a form of nondeterminism is not handled by our notion of effective plan, and we make the assumption that the specification Σ does not give rise to implicit nondeterminism.[1] Indeed, it may be possible that a conditional plan, dealing with this form of nondeterminism, exists in such cases. However, in the present work we are not interested in these situations, since we want to enforce non-sensing actions to be deterministic (although context-dependent) actions.

Reasoning method To the aim of generating conditional plans with concurrent execution of actions in the proposed framework, we extend the algorithm described in [De Giacomo *et al.*,1997b] for computing the *first-order extension* (FOE) of an $\mathcal{ALCK}_{\mathcal{NF}}$ knowledge

[1] It can be effectively checked whether Σ satisfies such a property, by verifying, in the algorithm reported in Figure 1, whether there exist two (or more) different default propagations of properties in the generation of the successor state.

base $\Sigma = \Gamma_S \cup \Gamma_P \cup \Gamma_E \cup \Gamma_{DFR} \cup \Gamma_{EFR} \cup \Gamma_I$ containing the formalization of the dynamic system in the terms described above. Informally, given a specification Σ and a goal G, $FOE(\Sigma, G)$ is an $\mathcal{ALC}$ knowledge base which consists of: (1) the static axioms in Γ_S; (2) the specification of the initial state (the assertions on *init* in Γ_I) augmented by the assertions which are consequences (up to renaming of individuals) of the epistemic sentences in Σ. The FOE of Σ provides a unique characterization of the knowledge that is shared by all the models of Σ, which is relevant with respect to the planning problem for the goal G.

The FOE of Σ for G is computed by the algorithm shown in Fig. 1. Roughly speaking, the algorithm, starting from the initial state *init*, applies to each state which does not satisfy the goal G the rules in the sets Γ_E, Γ_{DFR}, and Γ_{EFR} which are triggered by such a state, generating a new successor state, unless one with the same properties has already been created. The effects of the epistemic axioms are thus computed, and a non-epistemic "completion" of the knowledge base is obtained. In the algorithm,

$$\text{CONCEPTS}(\Gamma_S \cup \mathcal{A}, s) = \{C \mid \Gamma_S \cup \mathcal{A} \models C(s)\}$$

denotes the set of concepts that are *valid* for the explicitly named individual s, occurring in the set of instance assertions $\mathcal{A}$, wrt the $\mathcal{ALC}$ knowledge base $\Gamma_S \cup \mathcal{A}$, and $\text{POST}(s, R, \Gamma_S \cup \mathcal{A}, \Gamma_x) = \{D \mid \mathbf{K}C \sqsubseteq \forall \mathbf{K}R.\mathbf{K}D \in \Gamma_x \text{ and } \Gamma_S \cup \mathcal{A} \models C(s)\}$ denotes the effect of the application of all the triggered rules belonging to the set Γ_x involving the action R in the state s, namely the set of postconditions (concepts) of the rules in Γ_x which are triggered by s.

With respect to the algorithm described in [De Giacomo *et al.*,1997b] the following new issues have to be taken into account: (i) adding new transitions representing concurrent actions; (ii) verifying consistency of concurrent action execution; (iii) applying the new form of frame axioms. Hence the algorithm, for each subset of the set of possible actions in a given state s, attempts to create a new successor state with respect to the concurrent action. To this end, effect axioms are first applied for each single action, then consistency of the union of the sets of axioms obtained for each single action is verified.

It can be shown that the FOE is unique, that is, every order of extraction of states from the set ACTIVE_STATES and of the frame axioms from Γ_{DFR} and Γ_{EFR} produces the same set of assertions,[2] up to renaming of states. Moreover, the condition

[2]This property is a consequence of the assumption that Σ does not allow for implicit nondeterminism.

ALGORITHM FOE
INPUT: $\Sigma = \Gamma_S \cup \Gamma_P \cup \Gamma_E \cup \Gamma_{DFR} \cup \Gamma_{EFR} \cup \Gamma_I$, goal G
OUTPUT: $FOE(\Sigma, G)$ if Σ is consistent,
ERROR otherwise

PROCEDURE CreateNewState($s, \{R_1, \ldots, R_k\}$)
begin
  s' = NEW state name;
  $\mathcal{A}' = \mathcal{A} \cup \{R_1 \sqcap \ldots \sqcap R_k(s, s')\} \cup$
    $\{D(s') \mid 1 \leq i \leq k \wedge D \in \text{POST}(s, R_i, \Gamma_S \cup \mathcal{A}, \Gamma_E)\}$;
  if $\Gamma_S \cup \mathcal{A}' \not\models \bot$
  then begin
    for each axiom $\mathbf{K}C \sqsubseteq \forall \mathbf{K}R_i.\mathbf{A}\neg C \sqcup \mathbf{K}C \in \Gamma_{DFR}$
    such that $1 \leq i \leq k \wedge \Gamma_S \cup \mathcal{A} \models C(s)$ **do**
      if $\Gamma_S \cup \mathcal{A}' \not\models \neg C(s')$ **then** $\mathcal{A}' = \mathcal{A}' \cup \{C(s')\}$;
    for each axiom $\mathbf{K}C \sqsubseteq \forall \mathbf{K}R_i.\neg \mathbf{K}D \sqcup \mathbf{K}C \in \Gamma_{EFR}$
    such that $1 \leq i \leq k \wedge \Gamma_S \cup \mathcal{A} \models C(s)$ **do**
      if $\Gamma_S \cup \mathcal{A}' \models D(s')$ **then** $\mathcal{A}' = \mathcal{A}' \cup \{C(s')\}$;
    if there exists a state $s'' \in$ ALL_STATES
    such that
    CONCEPTS($\Gamma_S \cup \mathcal{A}, s''$)=CONCEPTS($\Gamma_S \cup \mathcal{A}', s'$)
    then $\mathcal{A} = \mathcal{A} \cup \{R_1 \sqcap \ldots \sqcap R_k(s, s'')\}$
    else begin
      $\mathcal{A} = \mathcal{A}'$;
      ACTIVE_STATES = ACTIVE_STATES $\cup \{s'\}$;
      ALL_STATES = ALL_STATES $\cup \{s'\}$
    end
  end
  else if $k = 1$ **then** EXIT WITH ERROR;
end;

begin
  $\mathcal{A} = \Gamma_I$; ACTIVE_STATES = $\{init\}$;
  ALL_STATES = $\{init\}$;
  repeat
    s = choose(ACTIVE_STATES); EXEC = $\emptyset$;
    if $\Gamma_S \cup \mathcal{A} \not\models G(s)$
    then begin
      for each ordinary action R_M **do**
        if there exists $\mathbf{K}C \sqsubseteq \exists \mathbf{K}R_M.\top \in \Gamma_P$
        such that $\Gamma_S \cup \mathcal{A} \models C(s)$
        then begin
          CreateNewState($s, \{R_M\}$);
          EXEC = EXEC $\cup \{R_M\}$
        end;
      for each sensing action R_S **do**
        if there exists $\mathbf{K}C \sqsubseteq \exists \mathbf{K}R_S.\top \in \Gamma_P$
        such that $\Gamma_S \cup \mathcal{A} \models C(s)$
          and $\Gamma_S \cup \mathcal{A} \not\models S(s)$
          and $\Gamma_S \cup \mathcal{A} \not\models \neg S(s)$
        then begin
          CreateNewState($s, \{R_S^+\}$);
          CreateNewState($s, \{R_S^-\}$);
          EXEC = EXEC $\cup \{R_S^+, R_S^-\}$
        end;
      for each subset $R_1, \ldots, R_k$ ($k \geq 2$) **of** EXEC **do**
        CreateNewState($s, \{R_1, \ldots, R_k\}$);
      ACTIVE_STATES = ACTIVE_STATES $-\{s\}$
    end
  until ACTIVE_STATES = $\emptyset$;
  return $\Gamma_S \cup \mathcal{A}$
end.

Figure 1: Algorithm computing $FOE(\Sigma)$

CONCEPTS($\Gamma_S \cup \mathcal{A}, s$) =CONCEPTS($\Gamma_S \cup \mathcal{A}', s'$) can be checked by verifying whether, for each concept C such that either $C(init) \in \Gamma_I$ or C is in the postcondition of some axiom in $\Gamma_E \cup \Gamma_{EFR} \cup \Gamma_{DFR}$, $\Gamma_S \cup \mathcal{A} \models C(s)$ iff $\Gamma_S \cup \mathcal{A}' \models C(s')$. Finally, it is easy to see that the total number of different instances of the above sets of concepts is finite (exponential in the number of axioms in Σ). Hence, the condition ACTIVE_STATES= $\emptyset$ is eventually reached, which implies that the algorithm terminates.

The notion of first-order extension constitutes the basis of a sound and complete planning method for the $\mathcal{ALCK}_{\mathcal{NF}}$ framework. More specifically, we show that the planning problem in Σ expressed by (4) can be reduced to an entailment problem for $FOE(\Sigma, G)$, by making use of the following translation function $\tau(\cdot)$.

Definition 3 *Let C be a concept expression representing a plan (i.e. belonging to the set $\mathcal{P}_C$). Then, $\tau(C)$ is the concept expression obtained as follows:*
(i) if $C = \mathbf{K}G$ then $\tau(C) = \mathbf{K}G$;
(ii) if $C = \exists \mathbf{K} R_{M_i}.C_1$ then $\tau(C) = \exists \mathbf{K} R_{M_i}.\tau(C_1)$;
(iii) if $C = \exists \mathbf{K} R_{S_i}.(\mathbf{K}S_i \sqcap C_1) \sqcup (\mathbf{K}\neg S_i \sqcap C_2)$ then $\tau(C) = \exists \mathbf{K} R^+_{S_i}.\tau(C_1) \sqcap \exists \mathbf{K} R^-_{S_i}.\tau(C_2)$;
(iv) if $C = \exists \mathbf{K}(R_{M_1} \parallel \ldots \parallel R_{M_k} \parallel R_{S_1} \parallel \ldots \parallel R_{S_l}).(\mathbf{K}S_{i_1} \sqcap \ldots \sqcap \mathbf{K}S_{i_j} \sqcap \mathbf{K}\neg S_{i_{j+1}} \sqcap \ldots \sqcap \mathbf{K}\neg S_{i_l} \sqcap C_1)$ then $\tau(C) = \exists \mathbf{K}(R_{M_1} \parallel \ldots \parallel R_{M_k} \parallel R^+_{S_{i_1}} \parallel \ldots \parallel R^+_{S_{i_j}} \parallel R^-_{S_{i_{j+1}}} \parallel \ldots \parallel R^-_{S_{i_l}}).\tau(C_1)$.

Theorem 4 *Let $C \in \mathcal{P}_C$. Then, $\Sigma \models C(init)$ iff $FOE(\Sigma, G) \models \tau(C)(init)$.*

In this way the planning problem consists of generating a constructive proof of the logical entailment $FOE(\Sigma, G) \models \Pi_G(init)$, where Π_G is a concept expression representing a plan for achieving the goal G from the initial situation *init*.

Planner implementation The framework and the algorithm described in the previous sections have been actually implemented and used to describe the knowledge of actual mobile robots embedded in real environments. The current implementation of the planner is built on top of the reasoning services provided by the well-known DL system CLASSIC [Borgida *et al.*,1989]. In particular, we make use of its built-in instance checking mechanism to check the validity of a concept in a state, and of triggering of rules to propagate effects. However, CLASSIC does not provide for a full implementation of the formalism. In particular, the epistemic operators **K** and **A** are handled by ad hoc attached procedures, the union operator is allowed only in the preconditions of the axioms, and negation is dealt with by the use of additional concepts.

The planning procedure, given an initial state and a goal, generates a conditional plan with concurrent actions that, when executed starting from the initial state, leads to a state in which the goal is satisfied. This procedure is based on the result of Theorem 4 and, specifically, aims at building a term (concept) Π_G such that the implication $FOE(\Sigma) \models \Pi_G(init)$ holds.

Such a plan could in principle be generated in two steps. First, the FOE of the knowledge base is generated (using the algorithm above): such a knowledge base can be seen as an action graph representing all possible plans starting from the initial state. Then, such a graph is visited, building a term (the plan) representing a tree in which: (i) sensing actions generate branches; (ii) concurrent actions are explicitly represented in the arcs; (iii) each branch leads to a state satisfying the goal. Obviously, several strategies can be applied to implement this method, and they are not addressed in this paper.

5 APPLICATION DOMAINS

In this section we present some examples of application of the above formalism, addressing its most relevant features. On the one hand, we refer to classical examples such as the vase-on-the-table [Lin and Shoham,1992; Gelfond and Lifschitz,1993; Giordano *et al.*,1998]. The formalization of this example, which is omitted here for lack of space, shows that, in our framework, the correct plan can be obtained by combining the use of epistemic and state constraints with the possibility of executing concurrent actions. On the other hand, we discuss a concrete application domain. The framework and the planning system presented above have been used to describe the knowledge of actual mobile robots embedded in dynamic office-like environments (see [Iocchi,1999] for a detailed description of the integration of the framework within the mobile robots) and in the scenario of robotic soccer players provided by RoboCup competitions. In the implementation, our planner generates Colbert activities (control programs) that can be directly executed by the Saphira system [Konolige *et al.*,1997], which controls the mobile base.

Before we introduce the examples, it is worth pointing out that our aim here is not to describe the full implementation, but rather to focus on the generation of plans that highlight the features of the formalism. In particular, we do not address here the execution mechanism and the problem of splitting the tasks to be accomplished between the reasoning component and the underlying control level. By exploiting the proposed approach, we have been able to formal-

$\mathbf{K}(\neg BallClose \sqcap \neg OpponentOnBall) \sqsubseteq$
$\quad \mathbf{K}(BallClose \sqcap OpponentOnBall) \sqsubseteq \exists \mathbf{K} tackle.\top$
$\mathbf{K}(\neg BallClose \sqcap OpponentOnBall) \exists \mathbf{K} intercept.\top$
$\mathbf{K}(BallClose \sqcap \neg OpponentOnBall) \sqsubseteq \exists \mathbf{K} kick.\top$
$\mathbf{K}(\neg BallClose \sqcap \neg OpponentOnBall) \sqsubseteq \exists \mathbf{K} goToBall.\top$

$\mathbf{K}\top \sqsubseteq \forall tackle.GoalProtected$
$\mathbf{K}\top \sqsubseteq \forall intercept.GoalProtected$
$\mathbf{K}\top \sqsubseteq \forall kick.(GoalProtected \sqcap \neg BallClose)$
$\mathbf{K}\top \sqsubseteq \forall goToBall.(GoalProtected \sqcap BallClose)$

$\mathbf{K} BallInLPS \sqsubseteq \exists \mathbf{K} senseBallClose.\top$
$\mathbf{K}\top \sqsubseteq \mathbf{K}(\forall senseBallClose.BallClose) \sqcup$
$\quad \mathbf{K}(\forall senseBallClose.\neg BallClose)$
$\mathbf{K} BallInLPS \sqsubseteq \exists \mathbf{K} opponentOnBall.\top$
$\mathbf{K}\top \sqsubseteq \mathbf{K}(\forall senseOpponentOnBall.OpponentOnBall) \sqcup$
$\quad \mathbf{K}(\forall senseOpponentOnBall.\neg OpponentOnBall)$

Figure 2: $\mathcal{ALCK}_{\mathcal{NF}}$-KB for Example 1.

ize at the logical level several situations arising in the RoboCup scenario and to generate, through the planner, a significant portion of the control programs that were executed on some of the soccer players that participated, within the ART team, in the RoboCup-99 F-2000 league [Nardi *et al.*,1999].

Example 1: Planning defensive actions In the first example we consider the case of two actions which sense fluents that can change by the effect of exogenous events (i.e., non-persistent fluents), and, therefore, may need to be executed concurrently.

Suppose that one wants to represent a defensive situation, where the goal of the robot is to protect its own goal. The robot can see the ball $BallInLPS$, and there are two sensing actions that allow to decide whether the ball is close ($BallClose$) and whether there is an opponent which is on the ball performing some offensive action ($OpponentOnBall$). Moreover, there are various actions that the robot can take to properly defend its goal: *tackle*, which is required when both robots are next to the ball; *intercept*, (i.e. put itself between the opponent and the goal) which must be preferred when the opponent is on the ball and the robot is not; *kick*, which can be accomplished when the robot is close to the ball and the opponent is not; *goToBall*, which should be chosen when the opponent is not on the ball. It is worth mentioning that there are real differences in the effectiveness of the above actions depending on the situation. For example when the ball is in between the robot and the opponent kicking (forward) is usually not very effective and may lead to a foul and a tackling action is more appropriate.

The knowledge base reported in Figure 2 reports precondition and effect axioms for moving and sensing actions. Given the initial situation $BallInLPS$ and the goal $GoalProtected$, the system generates the following conditional plan:

```
senseBallClose || senseOpponentOnBall;
if (BallClose and not OpponentOnBall)
  { kick; }
  else { if (BallClose and OpponentOnBall)
       { tackle; }
       else { if (not BallClose and OpponentOnBall)
            { intercept; }
            else { goToBall; }
```

Notice that if the two sensing actions are not executed concurrently, there is no plan, because neither $BallClose$ nor $OpponentOnBall$ persists after the execution of a sensing action. Furthermore, the goal is achieved whatever is the result of the sensing actions. It is worth emphasizing that, in general, the condition that the plan always leads to satisfy the goal is very strong. In fact, it is often the case that only certain paths in the action graph generated by computing the FOE lead to a state satisfying the goal. In such cases one can still extract from the FOE indications to drive the actions of the robot, but we cannot further exploit this issue here. Moreover, in practice, the execution of plans may fail due to the uncertainty and dynamics of the domain. The verification of such failures is embedded into the execution mechanism, whose details again cannot be provided here.

Example 2: Planning the pass In the RoboCup scenario, coordination among the robots is a critical issue, and we want to represent the robot's knowledge that allows for reasoning about concurrent actions performed by two or more robots. In particular, we shall refer to one of the challenges that have been proposed for coordinating players in RoboCup, namely the pass. We consider a situation in which two players $P1$ and $P2$ are executing an offensive action towards the opponent goal. Initially, $P1$ has possession of the ball ($BallPoss_1$) and can freely move ahead towards the opponent goal ($FreeAhead_1$). The idea is that $P1$ can move forward till it reaches a good position to shoot. At this point, it can check whether there is still the possibility to kick forward ($FreeAhead_1$ still holds) or else decide to pass to $P2$ to have a better chance to score. From this specification, it follows that each robot must be able to perform the following actions: *fwdKeepingBall* to move forward trying to keep ball possession, *positionForPass* to get to a proper position to receive a pass and conclude the action, *kick* for kicking the ball, *pass* to pass the ball, *receiveAndKick* to get the pass and conclude the action, and finally *senseFreeAhead* for detecting whether there are obstacles between itself and the opponent goal. We model

$\mathbf{K}(BallPoss_i \sqcap \neg FreeAhead_i \sqcap ShootPsn_j) \sqsubseteq$
$\quad \mathbf{K}(BallPoss_i \sqcap FreeAhead_i) \sqsubseteq \exists \mathbf{K} fwdKeepingBall_i.\top$
$\mathbf{K}(BallPoss_i \sqcap ShootPsn_i \sqcap FreeAhead_i) \sqsubseteq \exists \mathbf{K} kick_i.\top$
$\mathbf{K}(BallPoss_i \sqcap ShootPsn_j \sqcap \neg FreeAhead_i) \sqsubseteq \exists \mathbf{K} pass_i^j.\top$
$\mathbf{K}(BallClose_j \sqcap ShootPsn_j) \sqsubseteq \exists \mathbf{K} receiveAndKick_j.\top$
$\mathbf{K} BallPoss_i \sqsubseteq \exists \mathbf{K} positionForPass_j.\top$

$\mathbf{K}\top \sqsubseteq \forall fwdKeepingBall_i.(BallPoss_i \sqcap ShootPsn_i)$
$\mathbf{K}\top \sqsubseteq \forall kick_i.BallKicked$
$\mathbf{K}\top \sqsubseteq \forall pass_i^j.BallClose_j$
$\mathbf{K}\top \sqsubseteq \forall receiveAndKick_i.BallKicked$
$\mathbf{K}\top \sqsubseteq \forall positionForPass_i.ShootPsn_i$

$\mathbf{K}\top \sqsubseteq \exists \mathbf{K} senseFreeAhead_i.\top$
$\mathbf{K}\top \sqsubseteq \mathbf{K}(\forall senseFreeAhead_i.FreeAhead_i) \sqcup$
$\quad \mathbf{K}(\forall senseFreeAhead_i.\neg FreeAhead)$

$\mathbf{K} ShootPsn_i \sqsubseteq \forall \mathbf{K} pass_j^i.\mathbf{A} \neg ShootPsn_i \sqcup \mathbf{K} ShootPsn_i$
...

Figure 3: $\mathcal{ALCK}_{\mathcal{NF}}$-KB for Example 2.

this scenario through the $\mathcal{ALCK}_{\mathcal{NF}}$-KB reported in Figure 3, in which the subscript $i, j \in \{1,2\} \wedge i \neq j$ denotes actions and conditions concerning each robot.

Notice that the fluent representing the possession of the ball ($BallPoss$) does not persist, and, unless explicitly specified, as in the case of $fwdKeepingBall$, it must be verified by a sensing action; instead, $ShootPsn$ (representing the property of being in a good position for shooting) persists when a pass is executed, and persists by default during the execution of other actions.

Given the initial state in which $BallPoss_1 \sqcap FreeAhead_1$ holds and the goal $BallKicked$, we obtain the following plan.

```
senseFreeAhead1 || fwdKeepingBall1 || positionForPass2;
if (FreeAhead1)
  { kick1 }
  else { pass12; receiveAndKick2;}
```

We remark that the two actions $fwdKeepingBall_1$ and $senseFreeAhead_1$ must be executed concurrently, because $FreeAhead$ does not persist; moreover, the previous actions must be executed concurrently with $positionForPass_2$, because in any of the two corresponding sequences of moves of the two players, there is the possibility that an opponent moves the ball while $P1$ is waiting for the positioning of $P2$.

Plan execution. The introduction of a system that generates plans with concurrent actions requires the robotic architecture to be able to schedule and manage concurrent behaviors and to provide synchronization among such behaviors. Concurrent plans are actually executed on a single player by making use of the Saphira built-in mechnisms for activating concurrent behaviors and for monitoring their end before starting the next action in the plan. The execution of global concurrent plans (that concern more than one robot) is instead realized by means of explicit communication among players. Observe that in this case all the players share the same plan, and each player is able to identify the actions that must be executed. For example, the execution of the action $A_1||B_2$ is obtained by performing A on $P1$ and B on $P2$ and by a broadcast notification when actions terminate. In this way, all the robots involved in the global plan can detect when it is possible to start the next action in the plan.

6 CONCLUSIONS

In this paper we have presented a new framework for representing actions and generating plans, that can be executed by a (cognitive) mobile robot, from a declarative specification of the knowledge about the dynamic system. We believe that the most relevant aspects of the work presented in the paper are essentially two. The first one is the expressiveness of the proposed framework, which allows for representing epistemic states of the agent, sensing actions, primitive concurrent actions, state constraints. Many such features had been considered by previous work, however their combination gives rise to a new effective characterization of context-dependent actions and exogenous events. In addition, we have addressed plan generation in such a rich framework, which is often not considered by previous work, focusing mainly on the epistemological aspects of the formalization.

The second aspect worth remarking is that the framework has been implemented and used to generate plans for a concrete multi-robot scenario, provided by the RoboCup mid-size competition for robotic soccer. The examples presented should provide enough evidence that the representational features of the formalism are necessary in order to correctly model such a domain.

Several issues arising from the work presented in the paper could be investigated following the proposed approach. One interesting question, we believe, is how to embed the plans obtained from the specification into the execution mechanism. More precisely, the programs executed on the robots take the form of a transition graph. Even though we can currently generate portions of such a graph from the extracted plans, there are aspects of the actual execution, such as monitoring the failure and the termination of actions, that we deal with by ad-hoc ways, while a more systematic treatment would be needed.

References

[Baral and Gelfond, 1993] C Baral and M. Gelfond. Representing concurrent actions in extended logic programming. In *Proc. of the 13th Int. Joint Conf. on Artificial Intelligence (IJCAI'93)*, pages 866–871, 1993.

[Borgida *et al.*, 1989] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. CLASSIC: A structural data model for objects. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 59–67, 1989.

[De Giacomo and Lenzerini, 1994] Giuseppe De Giacomo and Maurizio Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI'94)*, pages 205–212. AAAI Press/The MIT Press, 1994.

[De Giacomo *et al.*, 1996] Giuseppe De Giacomo, Luca Iocchi, Daniele Nardi, and Riccardo Rosati. Moving a robot: the KR&R approach at work. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, 1996.

[De Giacomo *et al.*, 1997a] G. De Giacomo, H. J. Levesque, and Y. Lesperance. Reasoning about concurrent executions, prioritized interrupts, and exogenous action in the situation calculus. In *Proc. of the 15th Int. Joint Conf. on Artificial Intelligence (IJCAI'97)*, 1997.

[De Giacomo *et al.*, 1997b] Giuseppe De Giacomo, Luca Iocchi, Daniele Nardi, and Riccardo Rosati. Planning with sensing for a mobile robot. In *Proc. of the Fourth European Conf. on Planning (ECP'97)*, number 1348 in Lecture Notes In Artificial Intelligence, pages 156–168. Springer-Verlag, 1997.

[Donini *et al.*, 1997] Francesco M. Donini, Daniele Nardi, and Riccardo Rosati. Autoepistemic description logics. In *Proc. of the 15th Int. Joint Conf. on Artificial Intelligence (IJCAI'97)*, pages 136–141, 1997.

[Gelfond and Lifschitz, 1993] M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–322, 1993.

[Giordano *et al.*, 1998] L. Giordano, A. Martelli, and C. Schwind. Dealing with concurrent actions in modal action logic. In *Proc. Of European Conference on Artificial Intelligence (ECAI'98)*, 1998.

[Iocchi, 1999] Luca Iocchi. *Design and Development of Cognitive Robots*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1999.

[Konolige *et al.*, 1997] K. Konolige, K.L. Myers, E.H. Ruspini, and A. Saffiotti. The Saphira architecture: A design for autonomy. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(1):215–235, 1997.

[Lakemeyer and Levesque, 1998] Gerhard Lakemeyer and Hector J. Levesque. AOL: a logic of acting, sensing, knowing, and only knowing. In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'98)*, pages 316–327. Morgan Kaufmann, Los Altos, 1998.

[Lesperance *et al.*, 1998] Yves Lesperance, Kenneth Tam, and Michael Jenkin. Reactivity in a logic-based robot programming framework. In *Proceedings of AAAI98 Fall Symposium on Cognitive Robotics*, 1998.

[Levesque, 1996] Hector .J. Levesque. What is planning in presence of sensing? In AAAI Press/The MIT Press, editor, *Proc. of the 13th Nat. Conf. on Artificial Intelligence (AAAI'96)*, pages 1139–1149, 1996.

[Lifschitz, 1994] Vladimir Lifschitz. Minimal belief and negation as failure. *Artificial Intelligence*, 70:53–72, 1994.

[Lin and Reiter, 1994] F. Lin and R. Reiter. State constraints revisited. *J. of Logic and Computation*, 4(5):655–678, 1994.

[Lin and Shoham, 1992] F. Lin and Y. Shoham. Concurrent actions in the situation calculus. In *Proceedings of the 11th National Conference on Artificial Intelligence*, pages 590–595, 1992.

[Lin, 1995] F. Lin. Embracing causality in specifying the indirect effects of actions. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI'95)*, 1995.

[Lobo *et al.*, 1997] Jorge Lobo, Gisela Mendez, and Stuart R. Taylor. Adding knowledge to the action description language A. In *Proc. of the 14th Nat. Conf. on Artificial Intelligence (AAAI'97)*, pages 454–459, 1997.

[Mc Cain and Turner, 1995] N. Mc Cain and H. Turner. A causal theory of ramifications and qualifications. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI'95)*, 1995.

[Nardi *et al.*, 1999] Daniele Nardi, Giovanni Adorni, Andrea Bonarini, Riccardo Cassinis, Giorgio Clemente, Antonio Chella, Enrico Pagello, and Maurizio Piaggio. ART: Azzurra Robot Team. Available at `http://www.ida.liu.se/ext/RoboCup-99`, 1999.

[Pinto, 1998] Javier A. Pinto. Concurrent actions and interacting effects. In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'98)*, 1998.

[Reiter, 1991] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, 1991.

[Reiter, 1996] R. Reiter. Natural actions, concurrency and continuous time in the situation calculus. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 2–13, 1996.

[Rosenschein, 1981] Stanley J. Rosenschein. Plan synthesis: A logical perspective. In *Proc. of the 7th Int. Joint Conf. on Artificial Intelligence (IJCAI'81)*, pages 331–337, 1981.

[Scherl and Levesque, 1993] Richard Scherl and Hector J. Levesque. The frame problem and knowledge producing actions. In *Proc. of the 11th Nat. Conf. on Artificial Intelligence (AAAI'93)*, pages 689–695, 1993.

[Shanahan, 1997] M. Shanahan. Event calculus planning revisited. In *Proc. of the 4th European Conference on Planning (ECP'97)*, 1997.

[Thielscher, 1997] M. Thielscher. Ramification and causality. *Artificial Intelligence*, 89, 1997.

Appendix: $\mathcal{ALCK}_{\mathcal{NF}}$ semantics

Since the logic $\mathcal{ALCK}_{\mathcal{NF}}$ is a particular case of Lifschitz's logic *MKNF* [Lifschitz,1994], the semantics of $\mathcal{ALCK}_{\mathcal{NF}}$ is obtained by interpreting concepts and roles on possible-world structures corresponding to sets of first-order interpretations. Non-epistemic concepts are interpreted as subsets of a domain, while non-epistemic roles are interpreted as binary relations over such a domain. Formally, an *interpretation* $\mathcal{I} = (\Delta, \cdot^{\mathcal{V}(\mathcal{I})})$ consists of a *domain of interpretation* Δ, and an *interpretation function* $\cdot^{\mathcal{V}(\mathcal{I})}$ mapping each concept to a subset of Δ (we assume $\top^{\mathcal{V}(\mathcal{I})} = \Delta$ and $\bot^{\mathcal{V}(\mathcal{I})} = \emptyset$) and each role to a subset of $\Delta \times \Delta$ as follows (A denotes an atomic concept, C, possibly with a subscript, denotes a concept, P, possibly with a subscript, denotes an atomic role, and R denotes a role):

$$
\begin{aligned}
A^{\mathcal{V}(\mathcal{I})} &\subseteq \Delta \\
(\neg C)^{\mathcal{V}(\mathcal{I})} &= \Delta \setminus C^{\mathcal{V}(\mathcal{I})} \\
(C_1 \sqcap C_2)^{\mathcal{V}(\mathcal{I})} &= C_1^{\mathcal{V}(\mathcal{I})} \cap C_2^{\mathcal{V}(\mathcal{I})} \\
(C_1 \sqcup C_2)^{\mathcal{V}(\mathcal{I})} &= C_1^{\mathcal{V}(\mathcal{I})} \cup C_2^{\mathcal{V}(\mathcal{I})} \\
(\exists R.C)^{\mathcal{V}(\mathcal{I})} &= \{d \in \Delta \mid \exists d'.(d,d') \in R^{\mathcal{V}(\mathcal{I})} \\
&\quad \text{and } d' \in C^{\mathcal{V}(\mathcal{I})}\} \\
(\forall R.C)^{\mathcal{V}(\mathcal{I})} &= \{d \in \Delta \mid \forall d'.\ \text{if } (d,d') \in R^{\mathcal{V}(\mathcal{I})} \\
&\quad \text{then } d' \in C^{\mathcal{V}(\mathcal{I})}\} \\
P^{\mathcal{V}(\mathcal{I})} &\subseteq \Delta \times \Delta \\
(P_1 \sqcap \ldots \sqcap P_n)^{\mathcal{V}(\mathcal{I})} &= (P_1)^{\mathcal{V}(\mathcal{I})} \cap \ldots \cap (P_n)^{\mathcal{V}(\mathcal{I})}
\end{aligned}
$$

The semantics of epistemic sentences is based on the following *Common Domain Assumption*: in each possible-world structure, each interpretation is defined over the same, fixed, countable-infinite domain of individuals Δ. We define an *epistemic interpretation* as a triple $(\mathcal{I}, \mathcal{M}, \mathcal{N})$ where $\mathcal{I}$ is an interpretation and $\mathcal{M}, \mathcal{N}$ are sets of interpretations such that $\mathcal{I} = (\Delta, \cdot^{\mathcal{V}(\mathcal{I})})$ and all interpretations in $\mathcal{M}$ and $\mathcal{N}$ are defined over the domain Δ. Epistemic sentences are interpreted on epistemic interpretations as follows (again, we assume that $(\top)^{\mathcal{I},\mathcal{M},\mathcal{N}} = \Delta$ and $(\bot)^{\mathcal{I},\mathcal{M},\mathcal{N}} = \emptyset$):

$$
\begin{aligned}
(A)^{\mathcal{I},\mathcal{M},\mathcal{N}} &= A^{\mathcal{V}(\mathcal{I})} \\
(\neg C)^{\mathcal{I},\mathcal{M},\mathcal{N}} &= \Delta \setminus (C)^{\mathcal{I},\mathcal{M},\mathcal{N}} \\
(C_1 \sqcap C_2)^{\mathcal{I},\mathcal{M},\mathcal{N}} &= (C_1)^{\mathcal{I},\mathcal{M},\mathcal{N}} \cap (C_2)^{\mathcal{I},\mathcal{M},\mathcal{N}} \\
(C_1 \sqcup C_2)^{\mathcal{I},\mathcal{M},\mathcal{N}} &= (C_1)^{\mathcal{I},\mathcal{M},\mathcal{N}} \cup (C_2)^{\mathcal{I},\mathcal{M},\mathcal{N}} \\
(\exists R.C)^{\mathcal{I},\mathcal{M},\mathcal{N}} &= \{d \in \Delta \mid \exists d'.(d,d') \in (R)^{\mathcal{I},\mathcal{M},\mathcal{N}} \\
&\quad \text{and } d' \in (C)^{\mathcal{I},\mathcal{M},\mathcal{N}}\} \\
(\forall R.C)^{\mathcal{I},\mathcal{M},\mathcal{N}} &= \{d \in \Delta \mid \forall d'.\ \text{if } (d,d') \in (R)^{\mathcal{I},\mathcal{M},\mathcal{N}} \\
&\quad \text{then } d' \in (C)^{\mathcal{I},\mathcal{M},\mathcal{N}}\} \\
(\mathbf{K}C)^{\mathcal{I},\mathcal{M},\mathcal{N}} &= \bigcap_{\mathcal{J} \in \mathcal{M}} (C)^{\mathcal{J},\mathcal{M},\mathcal{N}} \\
(\mathbf{A}C)^{\mathcal{I},\mathcal{M},\mathcal{N}} &= \bigcap_{\mathcal{J} \in \mathcal{N}} (C)^{\mathcal{J},\mathcal{M},\mathcal{N}} \\
(P)^{\mathcal{I},\mathcal{M},\mathcal{N}} &= P^{\mathcal{V}(\mathcal{I})} \\
(P_1 \sqcap \ldots \sqcap P_n)^{\mathcal{I},\mathcal{M},\mathcal{N}} &= (P_1)^{\mathcal{I},\mathcal{M},\mathcal{N}} \cap \ldots \cap (P_n)^{\mathcal{I},\mathcal{M},\mathcal{N}} \\
(\mathbf{K}R)^{\mathcal{I},\mathcal{M},\mathcal{N}} &= \bigcap_{\mathcal{J} \in \mathcal{M}} (R)^{\mathcal{J},\mathcal{M},\mathcal{N}} \\
(\mathbf{A}R)^{\mathcal{I},\mathcal{M},\mathcal{N}} &= \bigcap_{\mathcal{J} \in \mathcal{N}} (R)^{\mathcal{J},\mathcal{M},\mathcal{N}}
\end{aligned}
$$

Intuitively, an individual $d \in \Delta$ is an instance of a concept C iff $d \in (C)^{\mathcal{I},\mathcal{M},\mathcal{N}}$ in the particular interpretation $\mathcal{I}$. An individual $d \in \Delta$ is an instance of a concept $\mathbf{K}C$ (i.e. $d \in (\mathbf{K}C)^{\mathcal{I},\mathcal{M},\mathcal{N}}$) iff $d \in C^{\mathcal{J},\mathcal{M},\mathcal{N}}$ for all interpretations $\mathcal{J} \in \mathcal{M}$. That is, d is "known" to be an instance of concept C if it belongs to the concept interpretation of each possible world in $\mathcal{M}$. An individual $d \in \Delta$ is an instance of a concept $\neg\mathbf{A}C$ (i.e. $d \in (\neg\mathbf{A}C)^{\mathcal{I},\mathcal{M},\mathcal{N}}$) iff $d \in \neg C^{\mathcal{J},\mathcal{M},\mathcal{N}}$ for at least one interpretation $\mathcal{J} \in \mathcal{N}$. Namely, an individual is "by default" not an instance of a concept if it belongs to the concept interpretation in at least one possible world of $\mathcal{N}$. Similarly, an individual $d \in \Delta$ is an instance of a concept $\exists\mathbf{K}R.\top$ iff there is an individual $d' \in \Delta$ such that $(d,d') \in R^{\mathcal{J},\mathcal{M},\mathcal{N}}$ for all $\mathcal{J} \in \mathcal{M}$.

An inclusion assertion $C \sqsubseteq D$ is satisfied in $(\mathcal{I}, \mathcal{M}, \mathcal{N})$ iff $(C)^{\mathcal{I},\mathcal{M},\mathcal{N}} \subseteq (D)^{\mathcal{I},\mathcal{M},\mathcal{N}}$, while an instance assertion $C(a)$ is satisfied in $(\mathcal{I}, \mathcal{M}, \mathcal{N})$ iff $a \in (C)^{\mathcal{I},\mathcal{M},\mathcal{N}}$ and $R(a,b)$ is satisfied in $(\mathcal{I}, \mathcal{M}, \mathcal{N})$ iff $(a,b) \in (R)^{\mathcal{I},\mathcal{M},\mathcal{N}}$. An inclusion $C \sqsubseteq D$ is *satisfied* by a structure $(\mathcal{M}, \mathcal{N})$ iff each interpretation $\mathcal{I} \in \mathcal{M}$ is such that $(\mathcal{I}, \mathcal{M}, \mathcal{N})$ satisfies $C \sqsubseteq D$. An assertion $C(a)$ (resp. $R(a,b)$) is satisfied by $(\mathcal{M}, \mathcal{N})$ iff each interpretation $\mathcal{I} \in \mathcal{M}$ is such that $(\mathcal{I}, \mathcal{M}, \mathcal{N})$ satisfies $C(a)$ (resp. $R(a,b)$). An $\mathcal{ALCK}_{\mathcal{NF}}$-knowledge base Σ is *satisfied* by a structure $(\mathcal{M}, \mathcal{N})$ iff each sentence (inclusion or membership assertion) of Σ is satisfied by $(\mathcal{M}, \mathcal{N})$.

Then, a preference semantics is defined over the structures satisfying Σ. Precisely, a set of interpretations $\mathcal{M}$ is a *model* for Σ iff the structure $(\mathcal{M}, \mathcal{M})$ satisfies Σ and, for each set of interpretations $\mathcal{M}'$, if $\mathcal{M} \subset \mathcal{M}'$ then $(\mathcal{M}', \mathcal{M})$ does not satisfy Σ.

The $\mathcal{ALCK}_{\mathcal{NF}}$-knowledge base Σ is *consistent* if there exists a model for Σ, *inconsistent* otherwise. Σ logically implies an inclusion assertion $C \sqsubseteq D$, denoted as $\Sigma \models C \sqsubseteq D$, if $C \sqsubseteq D$ is true in each model for Σ. Analogously, *instance checking* in Σ of an assertion $C(a)$ is defined as follows: $\Sigma \models C(a)$ iff $C(a)$ is satisfied by each model of Σ.

Satisfiability Algorithms and Finite Quantification

Matthew L. Ginsberg
CIRL
1269 University of Oregon
Eugene OR 97403–1269
ginsberg@cirl.uoregon.edu

Andrew J. Parkes
CIRL
1269 University of Oregon
Eugene OR 97403–1269
parkes@cirl.uoregon.edu

Abstract

This paper makes three observations with regard to the application of algorithms such as WSAT and RELSAT to problems of practical interest. First, we identify a specific calculation ("subsearch") that is performed at each inference step by any existing satisfiability algorithm. We then show that for realistic problems, the time spent on subsearch can be expected to dominate the computational cost of the algorithm. Finally, we present a specific modification to the representation that exploits the structure of naturally occurring problems and leads to exponential reductions in the time needed for subsearch.

1 Introduction

The last few years have seen extraordinary improvements in the effectiveness of general-purpose Boolean satisfiability algorithms. This work began with the application of WSAT [Selman et al., 1996] to "practical" problems in a variety of domains (generative planning [Kautz and Selman, 1992], circuit layout, and others) by translating these problems into propositional logic and then solving them using WSAT.

Although WSAT is unchallenged in its ability to find models for randomly generated satisfiable theories, systematic methods have closed the gap on theories corresponding to practical problems. The algorithm of choice appears to be RELSAT [Bayardo and Schrag, 1997], an extension of an idea from dynamic backtracking [Ginsberg, 1993]. RELSAT is systematic and can therefore deal with both satisfiable and unsatisfiable theories; for theories arising from practical problems, its performance is comparable to WSAT's.

The applicability of propositional algorithms to practical problems is anything but straightforward, however. As the problems become more interesting, their size grows enormously. A single axiom such as

$$\forall xyz.[\, a(x,y) \wedge b(y,z) \rightarrow c(x,z) \,] \qquad (1)$$

has d^3 ground instances if d is the size of the domain from which x, y and z are taken. Researchers have dealt with this difficulty by buying machines with more memory, or by finding clever axiomatizations for which ground theories remain manageably sized [Kautz and Selman, 1998a]. In general, memory and cleverness are both scarce resources and a more natural solution will need to be found.

We will call a clause such as (1) *quantified*, and assume throughout that the quantification is universal as opposed to existential, and that the domains of quantification are finite.

Quantified clauses are common in encodings of realistic problems, and these problems have in general been solved by converting quantified clauses to standard propositional formulae. The quantifiers are expanded first (possible because the domains of quantification are finite), and the resulting set of predicates is then "linearized" by relabelling all the atoms so that, for example, $a(2,3)$ might become v_{24}. The number of ground clauses produced is exponential in the number of variables in the quantified clause.

Our primary goal in this paper is to work with the quantified formulation directly, as opposed to its much larger ground translation. Unfortunately, there are significant constant-factor costs incurred in doing so, since each inference step will need to deal with issues involving the bindings of the variables in question. Simply finding the value assigned to $a(2,3)$ might well take two or three times as long as finding the value assigned to the equivalent v_{24}. Finding all occurrences of a given literal can be achieved in the ground case

by simple indexing schemes, whereas in the quantified case this is likely to require a unification step. Such routine but essential operations can be expected to significantly slow the cost of every inference undertaken by the system.

The fundamental point of this paper is that while there are costs associated with using quantified axioms, there can be significant savings as well. These savings are a consequence of the fact that the basic inference loop of existing boolean satisfiability engines uses an amount of time that scales with the size of the theory; use of quantified axioms can reduce the size of the theory so substantially that the constant-factor costs can be overcome.

We will make this argument in three phases. First, we begin in the next section by reviewing the WSAT and Davis-Putnam search algorithms. In Section 3, we identify a computational subtask that is shared by these and all other existing algorithms. This subtask is NP-complete in a formal sense, and we call it *subsearch* for that reason. We explain why algorithms can be expected to encounter subsearch at each inference step, and rewrite the WSAT and Davis-Putnam algorithms to make their use of subsearch problems explicit.

In Section 4, we discuss other consequences of the fact that subsearch is NP-complete. Search techniques can be used to speed the solution of NP-complete problems, and subsearch is no exception. We will show that subsearch cost dominates the running time on realistic instances. We then show that quantified axiomatizations support the application of simple search techniques to the subsearch problem, and argue that realistic examples are likely to lead to subsearch problems that can be solved much faster by intelligent subsearch than by the methods used by existing implementations of search algorithms.

Section 5 presents experimental results comparing the performance of WSAT and a generalized version that is capable of dealing with non-ground axioms like (1). For small theories, the constant-factor costs of dealing with the variables cause "lifted WSAT" to be about 10 times slower than its ground version. As the theories get larger, however, the gap narrows and at the largest sizes for which both ground and lifted solvers can be run the overhead of lifting seems to be a factor of only 2. The lifted solver has the advantage that it can handle much larger theories that the ground solver.

Section 6 discusses related work. Future work is discussed in Section 7.

2 Searching for Models

To provide a context for the ideas we are about to present, we begin with brief descriptions of the WSAT and Davis-Putnam algorithms. Firstly, WSAT:

Procedure 2.1 (WSAT)

for $i := 1$ to MAX-TRIES
  $P :=$ a randomly generated truth assignment
  for $j := 1$ to MAX-FLIPS
    if P is a solution, **then return** P
    else $c :=$ a randomly selected unsatisfied clause
      foreach literal $l \in c$
        $B(l) =$ number of clauses that would break if were to flip l
      $m =$ minimum value of $B(l)$
      $L_m = \{ l \mid B(l) = m \}$
      if $m = 0$ **then** flip a random literal from L_0
      else
        with probability p flip a random literal in c
        else flip a random literal from L_m
      end if
    end if
  end for
end for
return failure

At each iteration, WSAT first checks to see if the problem has been solved. If not, it flips an atom either selected randomly from an unsatisfied clause or selected so as minimize the number of clauses that will change from satisfied to unsatisfied. (There are other variants of WSAT [McAllester et al., 1997, and others], however the differences will not be relevant here).

The Davis-Putnam algorithm takes a problem C and computes $\texttt{solve}(C, \emptyset)$, where `solve` works with a partial assignment P of values to atoms, attempting to extend P to a satisfying assignment using depth-first search and unit-propagation after each branch:

Procedure 2.2 (Davis-Putnam)

To compute $\texttt{solve}(C, P)$:

if $\texttt{unit-propagate}(P)$ fails
  return failure
else
  set $P := \texttt{unit-propagate}(P)$
  if P is a solution, **then return** P
  $v :=$ an atom not assigned a value by P
  if $\texttt{solve}(C, P \cup \{v = \texttt{true}\})$ succeeds
    return $\texttt{solve}(C, P \cup \{v = \texttt{true}\})$
  else
    return $\texttt{solve}(C, P \cup \{v = \texttt{false}\})$

A crucial, but time-consuming, subroutine is *unit propagation*:

Procedure 2.3 (Unit propagation)
To compute `unit-propagate`(P):

while there is a currently unsatisfied clause $c \in C$ that contains at most one unvalued literal **do**
  if every atom in c is assigned a value by P, **then return** failure
  else $a :=$ the atom in c unassigned by P
    augment P by valuing a so that c is satisfied
  end if
end while
return P

This procedure assigns forced values to atoms by finding unsatisfied clauses containing single unassigned atoms, and valuing them in the way that satisfies the clause. The valued atoms are then propagated further. If unit propagation reveals the presence of a contradiction, we return failure. If it turns P into a solution, we return that solution. Otherwise, we pick a branch variable v and try binding it to true and to false in succession. For example, if the given problem contained the clause $a \vee \neg b \vee c$ and a had been valued false and b true by P, we would conclude without branching that c must be true if the clause is to be satisfied. If there were an additional clause $a \vee \neg c$ in the problem, valuing c would value every atom in this clause without satisfying it, and unit propagation would return failure to indicate that a contradiction had been detected.

So far we have not specified how to select the branch variable v. Effective implementations [Crawford and Auton, 1996, Li and Anbulagan, 1997] pick v to maximize the number of unit propagations that occur after v is valued, since each unit propagation reduces the size of the residual problem. This is typically done by identifying a set of candidate branch variables v', (generally by counting the number of binary clauses in which each such variable appears), and then computing `unit-propagate`$(P \cup \{v' =$ `true`$\})$ and `unit-propagate`$(P \cup \{v' =$ `false`$\})$ to see how many unit propagations actually take place. In practice, some effort is also made to order the values for v in a way that is likely to lead to a solution quickly.

3 Subsearch

Each iteration of either the WSAT or Davis-Putnam algorithms involves a search through the original theory for clauses that satisfy some numeric property. In WSAT, when we want to flip the atom a that minimizes the number of newly unsatisfied clauses, we need to count the number of clauses that contain a single satisfied literal and include a in its current sense; these are the clauses that will become unsatisfied when we flip a.

In unit propagation, used both by the main loop in Davis-Putnam and by the step that selects a branch variable, we need to find the currently unsatisfied clauses that contain precisely zero or one unvalued literals, and no satisfied literals. Even determining if a solution has been found involves checking to see if any clauses remain with zero satisfied literals.

All of these tasks can be rewritten using the following:

Definition 3.1 *Suppose C is a set of quantified clauses, and P is a partial assignment of values to the atoms in those clauses. We will denote by $S(C, P, u, s)$ the set of ground instances of C that have u literals unvalued by P and s literals satisfied by the assignments in P.*

We will say that the checking problem *is that of determining whether $S(C, P, u, s) \neq \emptyset$. By a* subsearch problem, *we will mean an instance of the checking problem, or the problem of either enumerating $S(C, P, u, s)$ or determining its size.*

What is the complexity of the checking problem? For simplicity, let us first consider the checking problem for $S(C, P, 0, 0)$. Quantified clauses are handled separately and so we will just consider a single clause c. An element of $S(\{c\}, P, 0, 0)$ corresponds to giving a value to each of the universally quantified variables in c in such a way that *every* literal in c is valued false by P. As an example consider (1) again, that is, suppose

$$c \;=\; \forall xyz.[\, a(x,y) \;\wedge\; b(y,z) \rightarrow c(x,z)\,] \qquad (2)$$

Finding $S(\{c\}, P, 0, 0)$ corresponds to finding values for x, y and z such that

$$\begin{aligned} a(x,y) \quad & \wedge \\ b(y,z) \quad & \wedge \\ \neg\, c(x,z) & \end{aligned}$$

Now, look on the literal $a(x, y)$ as a constraint on the tuple (x, y) with the allowed tuples being just those for which $a(x, y)$ is valued true by P. Similarly, the other two literals give constraints on (y, z) and (x, z). With this viewpoint, finding relevant values for the variables x, y and z corresponds to solving a Constraint Satisfaction Problem (CSP) with 3 constraints.

More generally, given a clause c we can produce an associated CSP that we dub the "sub-CSP" of the clause. Each universally quantified variable becomes a variable of this sub-CSP. Each literal l of c becomes a constraint on the variable contained in l, the allowed values being just those that cause l to be false in P.

A satisfying assignment of this sub-CSP directly corresponds to value assignments for the universally quantified variables such that every literal is false, and hence to an unsatisfied ground clause from c. That is, solutions of the sub-CSP from c correspond to elements of $S(\{c\}, P, 0, 0)$. Note that the sub-CSP is generally not a binary CSP. Also, the constraints, though not the constraint graph, depend on the current assignment P.

The conversion from a clause to a CSP did not impose any restrictions on the clause converted, and the resulting sub-CSPs are not of any special form. However, solving CSPs is NP-complete and so the following theorem should not be a surprise.

Theorem 3.2 *For fixed u and s, the checking problem is NP-complete.*

Proof. Checking is in NP, since a witness that $S(C, P, u, s) \neq \emptyset$ need simply give suitable bindings for the variables in each clause of C.

To see NP-hardness, we first consider the case $u = s = 0$. The argument is essentially a formalization of the connection to a sub-CSP as discussed above.

We reduce from a binary CSP, producing a single clause c and set of bindings P such that, with $C = \{c\}$, we have that $S(C, P, 0, 0) \neq \emptyset$ if and only if the original problem was satisfiable. The basic idea is that each variable in the constraint problem will become a quantified variable in c.

Suppose that we have a binary CSP Σ with variables $v_1, \ldots, v_n$ and with m binary constraints of the form $(v_{i1}, v_{i2}) \in \sigma_i$, where (v_{i1}, v_{i2}) is the pair of variables constrained by σ_i. For each such constraint, we introduce a corresponding binary relation $r_i(v_{i1}, v_{i2})$, and take c to be the single quantified clause

$$\forall v_1, \ldots, v_n. \bigvee_i r_i(v_{i1}, v_{i2})$$

For the assignment P, we set $r_i(v_{i1}, v_{i2})$ to false for all $(v_{i1}, v_{i2}) \in \sigma_i$, and to true otherwise.

Now note that since P values every instance of every r_i, $S(\{c\}, P, 0, 0)$ will be nonempty if and only if there is a set of values for v_i such that every literal in $\vee_i r_i(v_{i_1}, v_{i_2})$ is false. Since a literal $r_i(v_{i_1}, v_{i_2})$ is false just in the case the original constraint σ_i is satisfied, it follows that $S(\{c\}, P, 0, 0) \neq \emptyset$ if and only if the original theory Σ was satisfiable.

For the case of non-zero u or s we can reduce from the case of $S(\{c\}, P, 0, 0)$. Given a clause c then create u new (nullary) literals U_i and s new literals S_i. Define P' to be P together with assigning all the S_j to true but leaving all the U_i unvalued. Now define a new clause c' by

$$c' = c \vee \bigvee_i U_i \vee \bigvee_j S_j$$

Then c' has precisely u unvalued and s satisfied literals iff c has neither unvalued nor unsatisfied literals. Finding $S(\{c'\}, P', u, s)$ corresponds to finding $S(\{c\}, P, 0, 0)$. Hence, the checking problem for $S(C, P, u, s)$ is also NP-hard. ■

Our aim for the rest of this section is just to show that the algorithms can be recast in terms of the subsearch problems. For notational convenience in what follows, suppose that C is a theory and that l is a literal. By C_l we will mean that subset of the clauses in C that include l. If C contains quantified clauses, then C_l will as well; the clauses in C_l can be found by matching the literal l against the clauses in C.

A recasting of WSAT in the terms of Definition 3.1 follows:

Procedure 3.3 (WSAT)

```
for i := 1 to MAX-TRIES
  P := a randomly generated truth assignment
  for j := 1 to MAX-FLIPS
    if S(C, P, 0, 0) = ∅, then return P
    else c := a random selection from S(C, P, 0, 0)
      foreach literal l ∈ c
          B(l) = |S(C_¬l, P, 0, 1)|
      m = minimum value of B(l)
      L_m = { l | B(l) = m}
      if m = 0 then flip a random literal from L_0
      else
          with probability p flip a random literal in c
          else flip a random literal from L_m
      end if
    end if
  end for
end for
return failure
```

When selecting the literal l to flip, recall that l is currently false. Then $|S(C_{\neg l}, P, 0, 1)|$ is the number of ground clauses that will become unsatisfied when l is flipped from false to true. Just as WSAT appeals to subsearch, so does unit propagation:

Procedure 3.4 (Unit propagation)

while $S(C,P,0,0) \cup S(C,P,1,0) \neq \emptyset$ **do**
  if $S(C,P,0,0) \neq \emptyset$, **then** return failure
  else select $c \in S(C,P,1,0)$
    augment P so that c is satisfied
  end if
end while
return P

In Davis-Putnam, $S(C,P,0,0)$ corresponds to clauses that can never be satisfied hence generating a backtrack, and $S(C,P,1,0)$ corresponds to clauses generating forced values by unit propagation.

For the literal selection in Davis-Putnam, as mentioned in Section 2, it is often useful to build heuristic measures of the number of propagations that would be caused by setting a literal l to true. This can be approximated by counting the clauses that contain $\neg l$ and also are effectively binary. For this we need clauses containing precisely two unvalued literals one of them being the literal $\neg l$. In terms of subsearch this simply corresponds to the set $S(C_{\neg l},P,2,0)$, and the relevant subsearch problem is to find the size of this set, that is, to find $|S(C_{\neg l},P,2,0)|$ (see [Parkes, 1999] for more details).

It is important to realize that all we are doing here is introducing new notation; the algorithms themselves are unchanged. Also note that algorithmic details such as the settings of noise p [McAllester et al., 1997] and restart parameters (for WSAT) or the variable and value choice heuristics (for Davis-Putnam) are irrelevant to the general point that these algorithms depend on solving subsearch problems at each step.

In practice, efficient implementations of these algorithms typically compute $S(C,P,u,s)$ once during an initialization phase, and then update it incrementally. For example, suppose that a literal l is changed from unvalued to true, changing the partial assignment from P to $P' = P \cup \{l = \texttt{true}\}$. To compute the number of fully assigned but unsatisfied clauses after the update, we start with the number before, and add newly unsatisfied clauses (unsatisfied clauses previously containing the single unvalued literal $\neg l$). That is, a rule such as

$$S(C,P',0,0) = S(C,P,0,0) \cup S(C_{\neg l},P,1,0)$$

can be used for incremental maintenance of $S(C,P,0,0)$. Similar rules will apply to other incremental changes.

Reorganizing the computation in this way leads to substantial speedups, since the subsearch problem being solved no longer involves the entire theory C but smaller theories such as C_l or $C_{\neg l}$. The fact that these incrementation techniques are essential to the performance of modern search implementations provides evidence that the performance of these implementations is dominated by the subsearch computation time.

4 Subsearch and quantification

The arguments of the previous section suggest that the running time of existing satisfiability algorithms will be dominated by the time spent solving subsearch problems, since such time is potentially exponential in the size of the subtheory C_l when the literal l is valued, unvalued or flipped. In this section, we discuss the question of how much of a concern this is in practice, and of what (if anything) can be done about it. After all, one of the primary lessons of recent satisfiability research is that even problems that are NP-hard in theory tend strongly to be much easier on average than worst-case analyses would suggest.

Let us begin by noting that subsearch is *not* likely to be much of an issue for the randomly generated satisfiability problems that have been the focus of recent research and that have driven the development of algorithms such as WSAT. The reason for this is that if n is the number of clauses in a theory C and v is the number of variables in C, then random problems tend to be difficult only for very specific values of the ratio n/v. For 3-SAT (where every clause in C contains exactly three literals), difficult random problems appear at $n/v \approx 4.2$. For such a problem, the number of clauses in which a particular literal l appears will be small (on average $3 \times 4.2/2 = 6.3$ for random 3-SAT). Thus the size of the relevant subtheory C_l or $C_{\neg l}$ will also be small, and while subsearch cost still tends to dominate[1] the running time of the algorithms in question, there is little to be gained by applying sophisticated techniques to reduce the time needed to examine a relative handful of clauses.

For realistic problems, the situation is dramatically different. Here is an axiom from a logistics domain encoded in the "planning as satisfiability" or SATPLAN style [Kautz and Selman, 1992]:

[1] Determined using runtime profiling of the ground implementations.

$$\texttt{at}(o,l,t) \wedge \texttt{flight-time}(l,l',dt) \wedge \texttt{between}(t,t',t+dt) \rightarrow \neg\texttt{at}(o,l',t') \quad (3)$$

This axiom says that if an object o is at location l at time t and it takes time dt to fly from l to l', and t' is between t and $t+dt$, then o cannot be at l' at t'.

A given atom of the form $\texttt{at}(o,l,t)$ will appear in $|t|^2|l|$ clauses of the above form, where $|t|$ is the number of time points or increments and $|l|$ is the number of locations. Even if there are only 100 of each, the 10^6 axioms created seem likely to make computing $S(C_l,P,u,s)$ impractical.

Now suppose we distinguish between the work done directly by the search engine itself and the work done indirectly by solving subsearch problems. We contend that, generally speaking, the work done directly will scale more slowly with the problem size (e.g. as measured by the largest domain size) than the subsearch problems.

In the case of Davis-Putnam the search will have to maintain the partial assignment P and so be driven by the number of variables. However, the subsearch involves scanning the clauses, and so will be driven by their number. Generally, the number of clauses involved in the subsearch will be larger than the number of variables involved in any subsequent change to the state P; for the example above, the subsearch might need to consider the 10^6 clauses to derive just one forced literal. For WSAT, the search not only maintains P but also stores and manipulates the the set of unsatisfied clauses. However, in practice, the set of unsatisfied clauses again tends to be much smaller than the set of all clauses.

Search itself is then generally a matter of doing subsearch on clauses in order to make changes to some current state of the search engine and the size of this state is typically driven by the number of variables. However, on realistic problems, the number of clauses is likely to increase more rapidly with problem size than the number of variables. (This is clear for (3) and is also true for the axioms considered in the experimental work in Section 5.) Thus, it is indeed to be expected that the costs associated with subsearch will dominate the runtime for realistic problems. We have also confirmed this by doing runtime profiles on ground implementations.

Let us now examine the computation of $S(C_l,P,u,s)$ in a bit more detail. Suppose that we do indeed have an atom $a = \texttt{at}(O,L,T)$ for fixed O, L and T, and that we are interested in counting the number of unit propagations that will be possible if we set a to true. In other words, we want to know how many instances of (3) will be unsatisfied and have a single unvalued literal after we do so.

Existing implementations, faced with this problem (or an analogous one if WSAT or another approach is used), will note that they need now consider axioms of the form (3) with o, l and t fixed but with l', t' and dt allowed to vary. They examine every axiom of this form and simply count the number of possible unit propagations.

This approach is taken because existing systems use not quantified clauses such as that of (3), but the set of ground instances of those clauses. Computing $S(C,P,u,s)$ for ground C involves simply checking each axiom individually; indeed, once the axiom has been replaced by its set of ground instances, no other approach seems possible.

Set against the context of a quantified axiom, however, this seems inappropriate. Computing $S(C,P,u,s)$ for a quantified C by reducing C to a set of ground clauses and then examining each is equivalent to solving the original NP-complete problem by generate and test – and if there is one thing that we can state with confidence about NP-complete problems, it is that generate and test is not an effective way to solve them.

Returning to our example with $\texttt{at}(O,L,T)$ true, we are looking for variable bindings for l', dt and t' such that, amongst $\neg\texttt{flight-time}(L,l',dt)$, $\neg\texttt{between}(T,t',T+dt)$ and $\neg\texttt{at}(O,l',t')$, precisely two of these literals are false and the third is unvalued. Theorem 3.2 suggests that subsearch will be exponentially hard (with respect to the number of quantifiers) in the worst case, but what is it likely to be like in practice?

In practice, things are going to be much better. Suppose that for some possible destination l', we know that $\texttt{flight-time}(L,l',dt)$ is false for all dt except some specific value Δ. We can immediately ignore all bindings for dt except for $dt = \Delta$, reducing the size of the subsearch space by a factor of $|t|$. If Δ depended on previous choices in the search (aircraft loads, etc.), it would be impossible to perform this analysis in advance and thereby remove the unnecessary bindings in the ground theory.

Pushing this example somewhat further, suppose that Δ is so small that $T+\Delta$ is the time point immediately after T. In other words, $\texttt{between}(T,t',T+\Delta)$ will always be false, so that $\neg\texttt{between}(T,t',T+\Delta)$ will always be true and no unit propagation will be

possible for any value of t' at all. We can "backtrack" away from the unfortunate choice of l in our (sub)search for variable bindings for which unit propagation is possible. Such backtracking is not supported by the generate-and-test subsearch philosophy used by existing implementations.

This sort of computational savings is likely to be possible in general. For naturally occurring theories, most of the atoms involved are likely to be either unvalued (because we have not yet managed to determine their truth values) or false (by virtue of the closed-world assumption, if nothing else). Domain constraints will typically be of the form $a_1 \wedge \cdots \wedge a_k \rightarrow l$, where the premises a_i are atoms and the conclusion l is a literal of unknown sign. Unit propagation (or other likely instances of the subsearch problem) will thus involve finding a situation where at most one of the a_i is unvalued, and the rest are true.

What then should we expect for the average complexity of subsearch? Suppose that the largest domain size is D and we have clauses with up to U variables. For a ground solver, the checking problem takes time $O(D^U)$, exponential in the length of the clauses. How much better can we expect to do using intelligent subsearch? To answer this fully would require knowledge of the distribution of subsearch problems arising from a given set of clauses and search engine. We do not have this information, however, suppose that the distribution of subsearch problems were sufficiently random that the work on phase transitions [Mitchell et al., 1992, Crawford and Auton, 1996, and others] were relevant. Such work shows that the hardest problems occur in critical regions where problems are changing from "mostly satisfiable" to "mostly unsatisfiable". However, even in such "hardest regions" experiments show that costs are far below generate-and-test. When the number of variables U is not too small, we can at least expect intelligent subsearch to reduce the complexity to $O(D^{\alpha U})$ for some $\alpha < 1$. Furthermore, if the typical subsearch instance is not at a hard phase transition then maybe we can do a lot better. In fact, given the arguments above it is not unreasonable to hope that the average cost complexity of subsearch will often be reduced to polynomial rather than exponential dependence on U.

5 Experimental results

To examine the impact of our ideas in an experimental setting, we compared the performance of ground and lifted solvers on problems of a variety of sizes from a simple logistics planning domain [Kautz and Selman, 1998a, and others]. Note that our goal is not to measure time to solution, but the basic step rate of the underlying search engine. The point is to measure the overheads from lifting and see whether or not they will swamp the so-far-theoretical savings. (More details of the experimental methods and results can be found elsewhere [Parkes, 1999].)

The domain consists of objects and airplanes; the locations are given as a function of time by means of predicates such as $\texttt{aboard}(o, a, t)$ for an object o, airplane a and time t. The only actions are loading or unloading objects onto planes and flying the planes. Unlike the example of the last section, all flights are assumed to be of unit duration. We have axioms such as

$$\forall aboi.\quad a = b \vee \neg\texttt{aboard}(o, a, i) \vee \neg\texttt{aboard}(o, b, i) \tag{4}$$

$$\forall aoci.\quad \neg\texttt{at}(o, c, i) \vee \neg\texttt{aboard}(o, a, i+1) \vee \texttt{plane-at}(a, c, i) \tag{5}$$

The first is a consistency condition on the state; the second requires the relevant plane to be at the airport when loading an object.

The experiments themselves used WSAT to solve problems that use N aircraft to redistribute $O(N)$ objects that were initially located at random among $O(N)$ cities. In the problems we use here the length of optimal plans is independent of N. The number of variables is $O(N^2)$ and the number of clauses is $O(N^3)$.

We also simplified the theories before presenting them to WSAT, propagating unit literals and unit propagating to completion. For the case of lifted axioms we used a lifted implementation of propagation (however, the incorporation of this into a lifted version of Davis-Putnam is not yet completed). This simplification produces somewhat smaller theories. The ground solver can exploit this by entirely removing forced variables and satisfied clauses. However, in the lifted case such removals are not done as they would require breaking up the simple quantified structure. For example, if the simplification sets $\texttt{aboard}(0, 0, 0)$ to false then the ground case can remove that literal and associated clauses entirely, whereas in the lifted case we still store the literal (keeping it fixed at false) and the full set of axioms. This effect tends to favor the ground axiomatizations as opposed to lifted ones.

The lifted solver takes lifted clauses as input and solves the subsearch problems using a straightforward depth-first search over the values of the universally-quantified variables, pruning the (sub)-search when possible as

discussed in the previous sections. All experiments were run under Linux on a 400 MHz Pentium II with 256MB of memory.

Let us first consider the time taken to initialize WSAT by counting the numbers of unsatisfied clauses after the initial variable assignment. Figure 1 presents initialization times for the ground and the lifted solver with and without the crucial pruning enabled. We remark that we did make one modification to standard WSAT. Normally, WSAT initially assigns literals true or false with equal probability. Instead, for both lifted and ground solvers, we bias the assignment towards setting literals to false. This bias is natural because any solution of the axioms will have a similar bias; it also reduces the time spent by WSAT in its initial hill-climbing phase and can reduce the total number of flips needed.

Except for small values of N ground WSAT scales as $O(N^3)$, predictably proportional to the number of clauses, whereas the lifted scales as $O(N^2)$. To see that this is to be expected consider an axiom such as (5) (axioms similar to this dominate the runtime). The subsearch for unsatisfied clauses can be terminated whenever $\mathtt{at}(o, c, i)$ is false. This pruning is likely to be common because of the weighting of the initial assignment, or more generally because objects should only be in one city. Hence, the subsearch involving this axiom is reduced from size *ocia* to size *oci*. The former is of size $O(N^3)$, while *oci* is only of size $O(N^2)$ (recall that the number of time points is independent of N). Solving the subsearch problem efficiently improves the complexity of initialization.

We next consider flip performance. For this we consider two implementations of WSAT: WSAT(U) and WSAT(UB). Both of these store and maintain the set of Unsatisfied clauses. WSAT(UB) also maintains an integer, the Breakcount, for each literal. The breakcount of a variable is the number of clauses that would become unsatisfied if that variable were to be flipped. Storage of the breakcount is used in the ground code distributed from AT&T[2], and generally improves the fliprate. Note that the version we used for the initialization experiments was WSAT(U). Initializing the breakcounts is more complex than initializing the unsatisfied clauses, and so, for the initial set of clauses we consider, it turns out that intelligent subsearch would not show a complexity reduction. However, see Section 5.1 for an axiom where even WSAT(UB) shows gains.

Results are in Figure 2. The ground lines terminate at $N = 40$ because beyond that point memory usage, growing as $O(N^3)$ for the ground solver, exceeds physical memory. The current version of the lifted solver runs out of memory at about $N = 300$, corresponding to billions of ground clauses.

Ground WSAT outperforms the lifted version but the difference drops rapidly as the size of the problem increases. More precisely, for WSAT(UB) we expect that the time per flip should be linear in the problem size N. Experimentally, the leading terms are $7.2N$ μsec/flip for the lifted solver as compared to about $3.5N$ for the ground solver: for large problems the lifted solver is about twice as slow. Note that the overheads tend to be worse on the smaller problems; perhaps on large problems the lifted solver obtains more advantages from being able to store information in the physical cache.

It is important to note here that the axiomatization used to encode the domain has a structure that should show *no* gain from pruning. Had the underlying theory allowed efficient solution of subsearch problems, we could expect better scaling properties, and that the better scaling would easily overcome the small constant overhead.

5.1 Extended axioms

Earlier, we mentioned that WSAT(UB) would not achieve any savings from intelligent subsearch on the initial set of axioms. Here we give an example of an axiom for which even WSAT(UB) can achieve significant savings:

$$\begin{gathered} i+1 < k \ \wedge\ c \neq d \rightarrow \\ [(\ \mathtt{at}(o,c,i)\ \wedge\ \mathtt{plane\text{-}at}(a,c,i)\ \wedge \\ \mathtt{at}(o,c,k)\ \wedge\ \mathtt{plane\text{-}at}(a,d,k)\ )\ \rightarrow \\ \mathtt{aboard}(o,a,i+1)\ ] \end{gathered} \tag{6}$$

This says that if a plane is going to the destination of a package then the package should be loaded immediately. This axiom contains $O(N^4)$ ground clauses, which hence dominates the initialization time for ground WSAT(UB). However, the expected initialization time for lifted WSAT(UB) is $O(N^3)$, and that for WSAT(U) is just $O(N^2)$. Experimental results for initialization for this axiom are given in Figure 3.

5.2 Comments

The lifted solver currently uses only simple methods, and we expect it can be improved in a number of ways, for example

[2]For example, `http://www.research.att.com/~kautz`

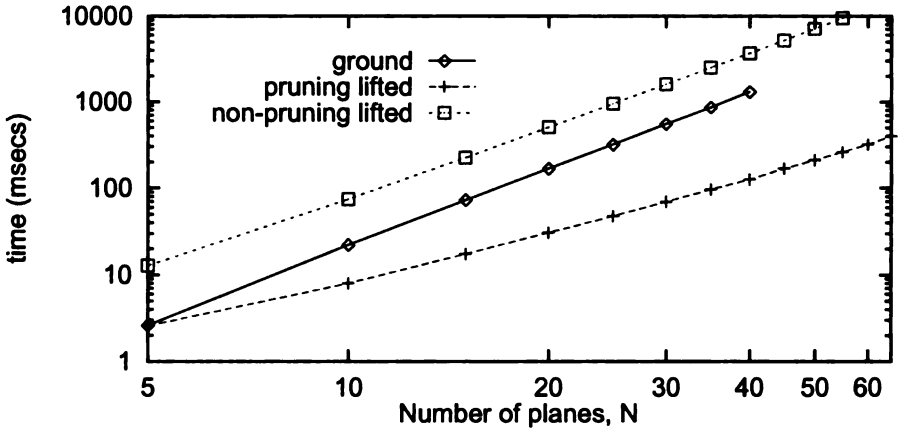


Figure 1: CPU time needed, during initialization in WSAT, to find the set of unsatisfied clauses.

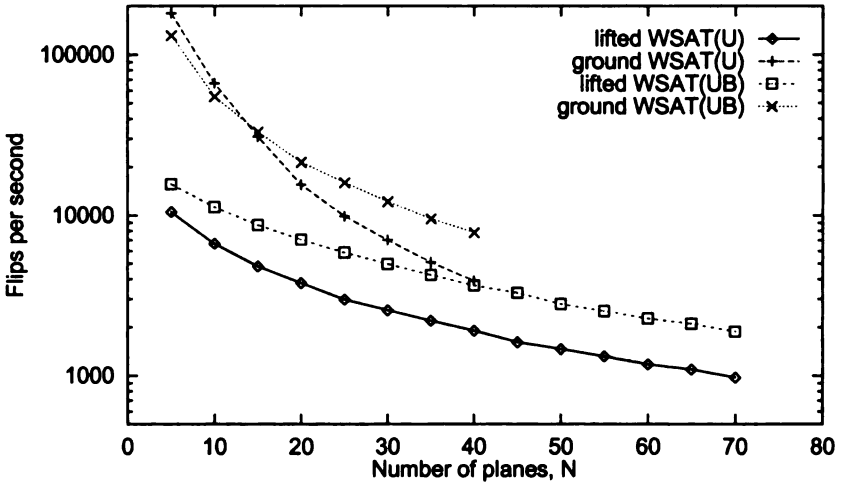


Figure 2: Flips per second for various versions of WSAT. The ground lines terminate because the physical memory of the machine is exhausted. "U" means that only unsatisfied clauses are cached. "UB" means that both unsatisfied clauses and breakcounts are cached.

- Do better subsearch. Currently, the subsearch only uses depth-first search with a static ordering of the variables. There are many other intelligent search methods that might do better. For example, we might exploit the structure of the sub-CSP [Parkes, 1999].

- Improve the state storage. Currently the state is stored as explicit multi-dimensional arrays of true, false or unset. In realistic instances the states are typically very structured; for example, biased towards false, or with many "adjacent" values being equal. In such cases, advanced data structures such as Boolean Decision Diagrams should be more compact and give faster access.

- Selectively ground clauses. For some lifted clauses we know that intelligent subsearch will never show any gains over the ground case. If such lifted clauses do not generate too many ground clauses, then we might selectively ground them out and avoid some of the overhead of lifting. Indeed, we might only keep as lifted the clauses that will gain from the intelligent subsearch or are simply too large to ground out.

We have seen that more complex constraints can make better use of intelligent subsearch: the larger the subsearch problem the larger the potential savings. Such larger axioms might arise in various ways. It is often possible to add clauses that are redundant in the sense of being logical consequences of the axioms but that can help the search by improving propagation properties. One might want to add soft constraints that merely improve the quality of the plans. One might

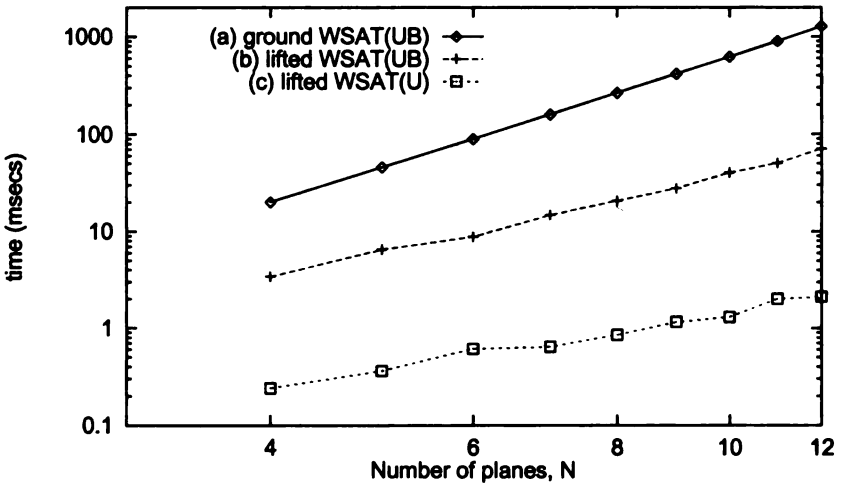


Figure 3: Experimental initialization times for WSAT on a large axiom (6). (a) Ground WSAT(UB). (b) Lifted WSAT(UB). (c) Lifted WSAT(U) Note that both axes have logarithmic scales: Line (a) scales as N^4, (b) scales as N^3, and (c) as N^2.

even add clauses designed to supply some domain-specific knowledge to guide the search engine itself [Kautz and Selman, 1998b]. All of these could easily lead to a large explosion in the size of the ground theory, and so require the use of a lifted solver.

6 Related Work

Other researchers have also examined the problems inherent in dealing with "global" constraints that are quantified over time or objects. They have typically assumed that the constraints involve many variables (polynomially many in the size of the problem), and that the constraints are "composite" in that they can be expressed using a large number of simpler constraints.

The composite nature of global constraints is most clear in constraint programming, where global constraints are often used because they can be implemented more efficiently than the equivalent set of more primitive constraints. A typical example is an `alldifferent` constraint on a set of variables X_i, which is equivalent to the $O(n^2)$ separate inequalities

$$\forall ij.\ [\, i = j \ \vee\ X_i \neq X_j \,]$$

The composite form can be implemented to run in time $O(n \log n)$ [Puget, 1998]. As an example, if all of the X_i are bound, we can sort their values to check for duplicates.

Lifted constraints are also composite, corresponding to the set of their ground instances. As with global constraints, we see implementation advantages in that the composite, lifted form can be implemented to run faster then the equivalent set of simpler, ground constraints.

There are major differences between our work and that of the global constraints community, however. These include the basic complexity of the constraints: lifted constraints are NP-complete with respect to the number of universal quantifiers, while global constraints such as `alldifferent` are in general polytime, corresponding to a polynomial number of primitive constraints. Checking an `alldifferent` constraint via enumeration will still only involve a polynomial number of tests (and does not seem like search); checking a lifted constraint via enumeration will need exponentially many tests (and does indeed seem like search). Perhaps more importantly, lifted constraints do not involve extensions beyond first-order syntax, and can therefore be combined effectively (e.g. by resolution) with other constraints in ways that `alldifferent` cannot be.

Sebastiani also considered the extension of GSAT to formula other than simple CNF [Sebastiani, 1994]. He did not consider the addition of quantifiers but instead allowed nesting of the standard boolean connectives. GSAT could then be implemented by a set of recursive rules to handle the nested formulas. His methods also obtained exponential speedups, and could handle boolean functions that would be exponentially large if reduced to CNF without the introduction of new variables.

His structuring of an algorithm such as GSAT in terms of specific subroutines such as one to count the number of unsatisfied clauses is similar to our split into search and subsearch. However, the important difference is that subsearch associated with universal quantifiers explicitly gives rise to independent search problems, and to associated benefits from intelligent search methods.

Also, in practice (at least in areas such as SATPLAN), it seems that universal quantifiers are rather more urgently needed than nesting (which can instead be eliminated by introducing new variables). Possibly, it would be interesting to combine our methods with those of Sebastiani. It is interesting that one way [Bedrax-Weiss et al., 1996] to convert POCL planning [McAllester and Rosenblitt, 1991] into SATPLAN results in an axiom with both quantifiers and nesting.

Finally, of course, our methods are hardly the only way to implement search on large problems; specialized code can handle domains of very large size. Our approach has the advantage of generality (and hence of maintainability). We believe it also illuminates general issues that would otherwise be obscured in the specialized code itself.

7 Conclusion and future work

Satisfiability algorithms have generally been developed against the framework provided by either random instances or, worse still, no unbiased source of instances at all. The algorithms themselves have thus tended to ignore problem features that dominate the computational requirements when they are applied to large real problems.

Principal among these appears to be *subsearch*, arising from the fact that most computational tasks involving realistic problems are NP-complete in the size of the quantified theory. We have argued that while subsearch is NP-complete in the size of the underlying theory it is amenable to the gains that can be expected from intelligent search methods, although existing satisfiability algorithms do not exploit this.

We described the reasons for this, and reported on the performance of a generalization of WSAT that is capable of dealing with universally quantified theories. If we consider large theories then, at worst, this generalization runs only modestly more slowly than the original ground WSAT, and in some cases it can use intelligent subsearch to achieve substantial savings. The generalization is also capable of solving problem instances for which ground WSAT would exhaust the physical memory available.

Unfortunately, similar ideas do not seem to apply to existential quantifiers. We do not know of anything better than just expanding out the existential (and indeed this is what is done in the current solver). In practice, this does not seem to a severe problem because existentials are less common than universals, and furthermore they are rarely nested. Consequently, expanding existentials generally does not have a disastrous effect on the size of the theory.

Grounding can still be better than lifting when the instances are hard and small so that the runtimes are long and memory is not a problem. Such a situation tends to occur when studying optimal planning using hand-crafted SATPLAN encodings that are selected to be small. Thus, lifting might well show easier gains if we are satisficing with large theories, or if we have to use automatically produced axioms for which the gains of hand-crafting, based on deep insight into the domain, might not all be available [Ernst et al., 1997]. Both of these conditions are likely to be common in practical situations. There is also the intriguing possibility that encodings could exist with good properties for search, and that would be totally impractical for use in ground solvers, but would still have good properties for lifting combined with intelligent subsearch. Note that a lot of effort has been spent [Kautz et al., 1996, and others] on generating encodings that produce small ground theories. Of necessity, properties of the encodings with respect to the search itself had to be secondary. Lifted solvers will allow the exploration of a much larger selection of encodings.

The key part of this work was the way that universal constraints are handled: that the theory was propositional was of lesser importance, Many of the ideas should be transferable to more general representation schemes. A natural first candidate would be to lift the linear 0-1 integer inequalities common in mathematical programming.

The most obvious extension to our work involves applying it to a systematic search engine. This should be straightforward for algorithms that do not involve any learning of nogoods. However, the most effective systematic algorithms do involve such learning, and exploiting lifting in this context appears to be more challenging than the WSAT work. We need to deal with the fact that the nogoods learned as the algorithm proceeds should be not the ground result of resolving ground database instances, but instead the result of unifying and then resolving the original quantified axioms. This leads to a variety of technical issues that have not yet been solved, and upon which we hope to

report in a subsequent paper.

Acknowledgements

Many of the issues and experiments described in this paper are also explored in the thesis of Parkes [Parkes, 1999].

This work was sponsored in part by grants from Air Force Office of Scientific Research (AFOSR), number F49620-92-J-0384, and Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Rome, NY, under agreements numbered F30602-95-1-0023, F30602-97-1-0294 and F30602-98-2-0181.

References

[Bayardo and Schrag, 1997] Bayardo, R. J. and Schrag, R. C. (1997). Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 203–208, Providence, RI.

[Bedrax-Weiss et al., 1996] Bedrax-Weiss, T., Jónsson, A. K., and Ginsberg, M. L. (1996). Unsolved problems in planning as constraint satisfaction. Available from http://www.cirl.uoregon.edu.

[Crawford and Auton, 1996] Crawford, J. M. and Auton, L. D. (1996). Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence*, 81:31–57.

[Ernst et al., 1997] Ernst, M. D., Millstein, T. D., and Weld, D. S. (1997). Automatic SAT-compilation of planning problems. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 1169–1176.

[Ginsberg, 1993] Ginsberg, M. L. (1993). Dynamic backtracking. *Journal of Artificial Intelligence Research*, 1:25–46.

[Kautz et al., 1996] Kautz, H., McAllester, D., and Selman, B. (1996). Encoding plans in propositional logic. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning*, pages 374–384, Boston, USA.

[Kautz and Selman, 1992] Kautz, H. and Selman, B. (1992). Planning as satisfiability. In *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI-92)*, pages 359–363. Vienna.

[Kautz and Selman, 1998a] Kautz, H. and Selman, B. (1998a). BLACKBOX: A new approach to the application of theorem proving to problem solving. In *Artificial Intelligence Planning Systems: Proceedings of the Fourth International Conference.* (Unpublished submission to planning competition).

[Kautz and Selman, 1998b] Kautz, H. and Selman, B. (1998b). The role of domain-specific knowledge in the planning as satisfiability framework. In *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems (AIPS'98)*, pages 181–189.

[Li and Anbulagan, 1997] Li, C. M. and Anbulagan (1997). Heuristics based on unit propagation for satisfiability problems. In *Proceedings of IJCAI-97*, pages 366–371.

[McAllester and Rosenblitt, 1991] McAllester, D. and Rosenblitt, D. (1991). Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 634–639.

[McAllester et al., 1997] McAllester, D., Selman, B., and Kautz, H. (1997). Evidence for invariants in local search. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 321–326.

[Mitchell et al., 1992] Mitchell, D., Selman, B., and Levesque, H. J. (1992). Hard and easy distributions of SAT problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 459–465.

[Parkes, 1999] Parkes, A. J. (1999). *Lifted Search Engines for Satisfiability.* PhD thesis, University of Oregon.

[Puget, 1998] Puget, J.-F. (1998). A fast algorithm for the bound consistency of alldiff constraints. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 359–366.

[Sebastiani, 1994] Sebastiani, R. (1994). Applying GSAT to non-clausal formulas. *Journal of Artificial Intelligence Research*, 1:309–314.

[Selman et al., 1996] Selman, B., Kautz, H., and Cohen, B. (1996). Local search strategies for satisfiability testing. In Johnson, D. S. and Trick, M. A., editors, *Cliques, Coloring and Satisfiability*, pages 521–531. American Mathematical Society. Proceedings of the second DIMACS Implementation Challenge, October 1993.

Desires and Defaults: A Framework for Planning with Inferred Goals*

Richmond H. Thomason
AI Laboratory
University of Michigan
Ann Arbor, MI 48109–2210
rich@thomason.org
http://www.eecs.umich.edu/~rthomaso/

Abstract

This paper develops a formalism designed to integrate reasoning about desires with planning. The resulting logic, BDP, is capable of modeling a wide range of common-sense practical arguments, and can serve as a more general and flexible model for agent architectures.

1 INTRODUCTION

In this paper, I will describe a formalism designed to integrate reasoning about desires with planning. You can motivate the ideas from two directions: (1) by reflecting on the need to extend planning formalisms to allow infererred goals, and (2) by explaining the need to extend a bare logic of belief and desire to a true system of practical reasoning by adding the capability to reason about actions. Let's follow the first path.

The AI paradigm for practical deliberation conceives of planning as a search for appropriate action sequences, given certain beliefs and desires—beliefs about the initial state of affairs, and desires about the outcome state. The formalization of the planning process focuses exclusively on means-end reasoning from set goals, and, in particular, it provides no provision for reasoning about goals. It provides no way to judge goals unworthy and to be discarded in the course of planning.

Commonsense planning is far less constrained; goals are frequently discarded in the course of many commonsense planning problems. Suppose, for instance, that I get down to planning a long-anticipated vacation. Prior to this stage I have formed specific desires about my itinerary. These are the things that I definitely would like to do, and these are the things that would become the goals of an AI-style planning solution to the problem. But as I work through the financial and scheduling details of the commonsense problem I discover that the travel cost is excessive, and the proportion of travel time is too high. The reasonable thing to do in that case is to adjust the itinerary by visiting fewer places, and that is what I do.

Goals come from desires, and desires can be impractical; they can also conflict with one another. (See, for instance, [1].) To arrive at practical recommendations, we may need to suspend some desires; but, as the travel example illustrates, we can't judge which desires to discard unless we integrate goal selection into the planning process.

The planning community is well aware of the need to be more flexible about goals. This awareness is evident in work on BDI architectures such as [10], and in work on plan execution monitoring and plan revision. However, the accepted formalization of planning provides no mechanism for incorporating this flexibility into the planning process itself, as it seems to be in commonsense examples.

To sum up: there are pretty compelling reasons for revising traditional planning formalisms to provide for reasoning about desires. Also, there are similarly compelling reasons for supplementing a nonmonotonic logic of beliefs and desires with a mechanism for planning. Both motives lead to the same sort of formalism: one that combines planning with general-purpose nonmonotonic reasoning about beliefs and desires. The logic BDP presented below in Section 3 is intended as a prototype system of this kind. I have purposely kept it as simple and generic as possible, in the hope that it can serve as a common starting point for developing more powerful and sophisticated formalisms for practical reasoning.

*Readers interested in elaborations of the ideas presented in this paper can consult the author's home page.

2 A LOGIC OF BELIEF AND DESIRE

2.1 TWO KINDS OF BELIEFS

In [6], a distinction is introduced between *skeptical* and *credulous* approaches to nonmonotonic reasoning. In general, formalisms that provide for defaults will allow sets of premises in which these defaults conflict; these cases are characterized by *multiple extensions*, theories representing different conclusion sets that could be reached from these premisses. In many reasoning applications, it is better to extract as much information from the premisses as possible, even at the risk of reaching some false conclusions. In these applications, a credulous strategy may be appropriate, in which the reasoner chooses one of many extensions.

In Raymond Reiter's default logic (see [12]) normal defaults (the only defaults I will consider) have the form $A \hookrightarrow C$. An axiomatization will consist of (1) a set M of formulas (the monotonic axioms), and (2) a set N of defaults (the nonmonotonic part of the axiomatization). A *theory* for an axiomatization $\langle M, N\rangle$ is a triple $\langle M, N, E\rangle$, where E is an extension of $\langle M, N\rangle$.[1] An extension E is closed under logical consequence. But, unlike the set of consequences of a set of monotonic axioms, it needn't be unique; a single axiomatization $\langle M, N\rangle$ can have many extensions.

The three components of a default theory can be mapped in a fairly natural way to attitudes of an ideal agent. A formula in M corresponds to an *immediate* belief, one that—at least, in the reasoning context under consideration—can't be retracted. A default in N corresponds to a *prima facie* belief, one that carries some conviction, but can be suspended in certain cases. A conclusion in E (at least, a conclusion that is not logically implied by M) corresponds to an *all-things-considered belief*, the outcome of a process of contradiction resolution and selection of competing defaults.

I am advocating, then, that some beliefs, at least, should be formalized as normal defaults.[2] I will call these beliefs, and their formalizations, *B-defaults*.

The distinction between *prima facie* and all-things-considered beliefs makes good intuitive sense. Take the following example.

Example 2.1. *Beliefs about the porch light.*

(i) I have a reason to believe the porch light is off, because I asked my daughter to turn it off.

(ii) I have a reason to believe the porch light is on, because the last time I saw it, it was on.

(iii) All things considered, I believe the porch light is off, because my daughter is pretty reliable.

In this example, *Prima facie* beliefs (i) and (ii) conflict with each other. The conflict is resolved by discarding (ii) and retaining (i) in reaching the all-things-considered belief (iii). Note that, like an intention, the all-things-considered belief that the light is off acts as a constraint on future deliberation; I will not form a plan to turn it off if I believe, all things considered, that it is already off.

2.2 WISHES AND WANTS

There are systematic similarities between beliefs and desires, which are reflected in the language used to describe them and in commonsense reasoning. Of special importance for motivating the formalism that I am aiming for here is an analogy between *prima facie* and all-things-considered beliefs on the one hand, and *wishes* and *wants* on the other.

Commonsense practical reasoning is concerned with the practicalization of desires. Immediate desires needn't be feasible, and typically will conflict with other immediate desires. We do not expect all of these wishes to survive as practical goals. The ones that do survive I will call *wants*.

This distinction seems to correspond to one important difference between the way 'wish' and 'would like' on the one hand and 'want' on the other are typically used. The following example shows that wishes can conflict with beliefs.

Example 2.2. *An infeasible wish.*

I'd like to take a long vacation.
I'd need to get time off from work to take a long vacation.
But: I can't get time off from work.

Also, wishes can conflict with each other, in light of background beliefs.

Example 2.3. *Conflicting wishes.*

I'd like to take a long vacation.
But: I'd like to save more money this year.

[1] A generalization of Reiter's definition of extension, for systems with defaults for both beliefs and desires, is given in Section 2, below.

[2] There are some similarities between this approach to belief and that of Veltman's Update Semantics. See [14].

And: I can't save more money this year and take a long vacation.

Finally, wishes can conflict with intentions, or more generally with adopted plans. This point is made by Michael Bratman, David Israel, and Martha Pollack. See [3, 2].

Example 2.4. *Conflicting wishes and intentions.*

I'd like to take a nap.
But: I intend to catch a plane.
So: I can't take a nap.

2.3 MODELING BELIEF AND DESIRES WITH DEFAULTS[3]

Exploiting the analogy developed in Section 2.2, I propose to use default rules to formalize both beliefs and desires. However, as we will see, it will be important to mark the difference between belief-based and desire-based defaults. I will use $A \overset{B}{\hookrightarrow} C$ for the first sort of default rule, and $A \overset{D}{\hookrightarrow} C$ for the second sort of default rule.

An axiomatization in the logic BD, or a *BD-basis,* will now consist of (1) a set M of formulas (the nonmonotonic axioms), (2) a set NB of B-defaults, and (3) a set ND of desire defaults, or *D-defaults.* A theory *induced by* a BD-basis $\langle M, NB, ND\rangle$ is a triple $\langle M, N, E\rangle$, where E is an extension of $\langle M, N\rangle$. Before defining the crucial notion of an extension, we will consider some motivating examples.

Example 2.5. *The Reasoning in Natural Language.*

1. Coffee is available.
2. I'd like decaf coffee if decaf coffee is available.
3. Decaf coffee must be available if coffee is available.
4. To have decaf coffee, I'll need to order decaf coffee.
5. *So:* I'll order decaf coffee.

Example 2.5, continued. *Formalizing the Premises*

1. $\top \overset{B}{\hookrightarrow}$ `Coffee-Available`
2. `Coffee-Available` $\overset{D}{\hookrightarrow}$ `Have-Decaf`
3. `Coffee-Available` $\overset{B}{\hookrightarrow}$ `Decaf-Available`
4. `Have-Decaf` $\overset{B}{\hookrightarrow}$ `Order-Decaf`

Example 2.5, continued. *Notes on the Formalization of Example 2.5*

[3]These ideas have been presented in several conference papers. See [13].

1. Premise 1, above, is a default, representing a guess. If Premise 1 had read `Coffee-Available` it would have represented an immediate, unretractable belief.
2. Intuitively, the premisses recorded in Example 2.5 should have only one reasonable conclusion. (Recall that conclusions are generated by a combination of beliefs and desires.) I should assume that coffee is available and that decaf coffee is available. I should want to have decaf, and I should order decaf. So the formal theory should deliver only one extension in this case, the one which is generated by the following choices:

   {`Coffee-Available`, `Decaf-Available`, `Have-Decaf`, `Order-Decaf`}
3. The above extension is the only one that is generated by Reiter's definition. (I am assuming here that the definition is applied in the simplest way, treating both sorts of defaults similarly.)
4. Premise 4, above, is a fairly crude way of compiling information that would be more appropriately be inferred as part of a planning process. The formalism of Section 3 will incorporate explicit end-means reasoning.

In the next example, imagine a hiking scenario.

Example 2.6. *The Reasoning in Natural Language.*

1. I think it's going to rain.
2. If it rains, I'll get wet.
3. I wouldn't like to get wet.
4. *So:* I'll get wet.

Example 2.6, continued. *Formalizing the Premises*

1. $\top \overset{B}{\hookrightarrow}$ `Rain`
2. `Rain` $\overset{B}{\hookrightarrow}$ `Wet`
3. $\top \overset{D}{\hookrightarrow} \neg$`Wet`

In the presence of Premise 1, there is a direct conflict between the defaults in Premises 2 and 3. The former premise represents a belief, the latter a desire.

In this case, there should be only one extension, which is generated by the following choices: {`Rain`, `Wet`} If I genuinely believe that it will rain, and that I will get wet if it rains, I should believe that I will get wet, regardless of my preferences or likings. To do otherwise would be to indulge in *wishful thinking.* Wishful thinking is incorrect, regardless of whether the beliefs are defeasible. To take another example, suppose that

I am not at home now, that I believe my umbrella is home now, and that I would like to have my umbrella in an hour. If wishful thinking were not prohibited, and waiting is available as an action, nothing would prevent a plan in which I do nothing, achieving my goal by mere wish fulfillment.

To put it another way, if practical reasoning is to be practical, wishes can't be fulfilled simply because they are wishes, but have to be achieved by feasible actions. It is belief that determines feasibility, not desire. So in a formalism that allows desire-based defaults, belief-based defaults have to take precedence over desire-based defaults in cases when there is any conflict between the two.

In the context of the formalism I am proposing, prioritization of belief over desire can be achieved by using a more or less standard account of prioritized defaults. See, for instance, [4]. Technical details are provided in Section 2.4, below.

With belief defaults prioritized over desire defaults, we obtain a single extension in Example 2.6, the one generated by the following choices: {Rain, Wet}.

The next example elaborates the hiking scenario of Example 2.6.

Example 2.7. *The Reasoning in Natural Language.*

1. I think it's going to rain.
2. If it rains, I'll get wet.
3. I wouldn't like to get wet.
4. If I get wet, I'll stay wet unless I give up and go home.
5. I wouldn't like to stay wet.
6. I wouldn't like to give up and go home.

Example 2.7, continued. *Formalizing the Premises*

1. $\top \overset{B}{\hookrightarrow} \texttt{Rain}$
2. $\texttt{Rain} \overset{B}{\hookrightarrow} \texttt{Get-Wet}$
3. $\top \overset{D}{\hookrightarrow} \neg\texttt{Get-Wet}$
4. $[\texttt{Get-Wet} \wedge \neg\texttt{Go-Home}] \overset{B}{\hookrightarrow} \texttt{Stay-Wet}$
5. $[\texttt{Get-Wet} \wedge \neg\texttt{Stay-Wet}] \overset{B}{\hookrightarrow} \texttt{Go-Home}$
6. $\top \overset{D}{\hookrightarrow} \neg\texttt{Stay-Wet}$
7. $\top \overset{D}{\hookrightarrow} \neg\texttt{Go-Home}$

In this case, there should be two extensions. The first is generated by the following choices:

$$\{\texttt{Rain}, \texttt{Get-Wet}, \texttt{Go-Home}, \neg\texttt{Stay-Wet}\}$$

The second is generated by the following choices:

$$\{\texttt{Rain}, \texttt{Get-Wet}, \neg\texttt{Go-Home}, \texttt{Stay-Wet}\}$$

These two extensions represent the two decision alternatives that the scenario of Example 2.7 makes available; on the one hand getting dry, but going home, and on the other continuing the hike, but staying wet.

In a purely epistemic version of default logic, multiple extensions represent equally reasonable alternatives, and the logic itself provides no way to choose between them. With desire-based defaults added to the mix, multiple extensions are still equally reasonable as far as the logic is concerned, but in many cases an agent will see some of these extensions as obviously preferable, and will have no difficulty in choosing among them. An agent who dislikes being wet much more than turning back will prefer the first extension in Example 2.7. These are the sorts of choices that numerical utilities are designed to resolve. I do not think that the logic should be expected to do more than to make the choices apparent. See Section 3.7 for further discussion of this issue.

2.4 CHARACTERIZING THE EXTENSIONS OF A BD-BASIS

Here is the idea behind the formal definition to be given below. A BD-extension E of a BD-basis $\langle M, NB, ND\rangle$ is a mininal first order theory that is closed under all the defaults that are applicable to it. Prioritization of B-defaults to D-defaults is ensured by allowing a D-default to be applicable to E only if there is no set of conflicting B-defaults.

In the following definitions, $S = \langle M, NB, ND\rangle$, and $\vdash$ is the consequence relation of first order logic; $\mathrm{Th}_{\mathrm{FOL}}(T) = \{A : T \vdash A\}$. Following Reiter, we define applicability relative to two parameters: a set T of premises and a "conjectured extension" T^* that is used to test consistency in applying rules.

Definition 2.1. *Applicability for B-defaults.*

A default rule $A \overset{B}{\hookrightarrow} C$ is applicable to T relative to T^*, where T and T^* are sets of formulas, iff (1) $T \vdash A$ and $T^* \nvdash \neg C$. $A \overset{B}{\hookrightarrow} C$ is *vacuously* applicable to T relative to T^* if it is applicable to T relative to T^* and $C \in T$.

Definition 2.2. *B-conflictedness for D-defaults.*

$A \overset{D}{\hookrightarrow} C$ is B-conflicted for T with respect to T^*, S iff for some $A_1 \overset{B}{\hookrightarrow} C_1, \ldots, A_n \overset{B}{\hookrightarrow} C_n \in NB$, $T \vdash A_i$ for all i, $1 \leq i \leq n$ and $T \cup \{C_1, \ldots, C_n\} \vdash \neg C$.

Definition 2.3. *Applicability for D-defaults.*

A default rule $A \overset{D}{\hookrightarrow} C$ of $\langle M, NB, ND\rangle$ is applicable to T, relative to T^* and S, if (1) $T \vdash A$ and $T^* \not\vdash \neg C$, and (2) $A \overset{D}{\hookrightarrow} C$ is not B-conflicted for T with respect to T^*. $A \overset{D}{\hookrightarrow} C$ is *vacuously applicable* to T relative to T^* if it is applicable to T relative to T^* and $C \in T$.

Definition 2.4. *BD-closure.*
T is BD-closed, relative to S, T^*, iff (1) $T = \mathrm{Th}_{\mathrm{FOL}}(T)$, (2) $M \subseteq T$, (3) for all $A \overset{B}{\hookrightarrow} C \in NB$, $C \in T$ if $A \overset{B}{\hookrightarrow} C$ is applicable to T relative to T^*, and (4) for all $A \overset{D}{\hookrightarrow} C \in ND$, $C \in T$ if $A \overset{D}{\hookrightarrow} C$ is applicable to T relative to T^*.

Definition 2.5. *BD-extension.*
E is a BD-extension of a BD-basis S iff (1) E is BD-closed, relative to S, E, and (2) for all E' such that E' is BD-closed, relative to S, E', we have $E' = E$ if $E' \subseteq E$.

BD-extensions can also be characterized in a more constructive way, by conjecturing an extension (a set T^*) and using this set for consistency checks in a proof-like process that applies defaults S successively to stages that begin with M; such a process yields a BD-extension if it produces T^* as its limit. The following definition invokes an "alphabetical" ordering of $NB \cup ND$. Any function from ω onto $NB \cup ND$ will serve the purpose.

Definition 2.6. *BD-proof process.*

$\mathcal{P}(S, T^*)$ is the sequence $\{T_i^{\mathcal{P}(S,T^*)} : i \in \omega\}$ defined as follows:

1) $T_0^{\mathcal{P}(S,T^*)} = M$.
2) $T_{i+1}^{\mathcal{P}(S,T^*)} = \mathrm{Th}_{\mathrm{FOL}}(T_i^{\mathcal{P}(S,T^*)} \cup \{C\})$ if there is a default in $NB \cup ND$ that is nonvacuously applicable to $T_i^{\mathcal{P}(S,T^*)}$ relative to S, T^*, where the alphabetically first such default has the form $A \overset{B}{\hookrightarrow} C$ or $A \overset{D}{\hookrightarrow} C$.
3) $T_{i+1}^{\mathcal{P}(S,T^*)} = T_i^{\mathcal{P}(S,T^*)}$ if no default in NB or ND is nonvacuously applicable to $T_i^{\mathcal{P}(S,T^*)}$ relative to S, T^*.

Definition 2.7. *Lim*$(\mathcal{P}(S, T^*))$
$\mathrm{Lim}(\mathcal{P}(S, T^*)) = \bigcup\{T_i^{\mathcal{P}(S,T^*)} : i \in \omega\}$.

In this version, I will simply state the following theorems without proof.

Theorem 2.1. Let E be a BD-extension of $S = \langle M, NB, ND\rangle$. Then $E = \mathrm{Lim}(\mathcal{P}(S, E))$.

Theorem 2.2. Let $E = \mathrm{Lim}(\mathcal{P}(S, E))$, where $S = \langle M, NB, ND\rangle$. Then E is a BD-extension of S.

In view of Theorem 2.2, we can show that T is a BD-extension by (1) using T for consistency checks in a default reasoning process from $\langle M, NB, ND\rangle$, (2) taking the limit T' of this process, and (3) verifying that in fact $T' = T$.

In Examples 2.5–2.7, some instances of informal reasoning involving beliefs and desires were formalized, along with remarks about the extensions that the examples seemed to require. The characterization of BD-extension provided by Definition 2.5 matches the requirements of these examples.

In particular, consider Example 2.6. This corresponds to the BD-basis $\langle M, NB, ND\rangle$, where:

$$M = \emptyset;$$
$$NB = \{\top \overset{B}{\hookrightarrow} \mathtt{Rain}, \mathtt{Rain} \overset{B}{\hookrightarrow} \mathtt{Wet}\};$$
$$ND = \{\mathtt{Rain} \overset{D}{\hookrightarrow} \neg\mathtt{Wet}\}.$$

It is straightforward to use Theorem 2.2 to show that $\{\mathtt{Rain}, \mathtt{Wet}\}$ is a BD-extension. The proof process generated by using $\mathrm{Th}_{\mathrm{FOL}}(\{\mathtt{Rain}, \mathtt{Wet}\})$ as a conjectured extension produces $\mathrm{Th}_{\mathrm{FOL}}(\{\mathtt{Rain}\})$ at the first step, $\mathrm{Th}_{\mathrm{FOL}}(\{\mathtt{Rain}, \mathtt{Wet}\})$ at the second step, and remains constant thereafter. Theorem 2.2 can also be used to show that the unwanted "wishful thinking" conclusion set, $\mathrm{Th}_{\mathrm{FOL}}(\neg\{\mathtt{Wet}\})$, is *not* a BD-extension. The proof process generated by using $\mathrm{Th}_{\mathrm{FOL}}(\neg\{\mathtt{Wet}\})$ as a conjectured extension produces $\mathrm{Th}_{\mathrm{FOL}}(\{\mathtt{Rain}\})$ at the first step, and remains constant thereafter. The B-default $\mathtt{Rain} \overset{B}{\hookrightarrow} \mathtt{Wet}$ cannot be applied, because the conclusion conflicts with the conjectured extension. The D-default $\mathtt{Rain} \overset{D}{\hookrightarrow} \neg\mathtt{Wet}$ cannot be applied, because it is B-conflicted.

In Section 3, we will need the following definitions.

Definition 2.8. *B-closure.*
T is B-closed, relative to $S = \langle M, NB\rangle$, T^*, iff (1) $T = \mathrm{Th}_{\mathrm{FOL}}(T)$, (2) $M \subseteq T$, and (3) for all $A \overset{B}{\hookrightarrow} C \in NB$, $C \in T$ if $A \overset{B}{\hookrightarrow} C$ is applicable to T relative to T^*.

Definition 2.9. *B-extension.*
E is a B-extension of $S = \langle M, NB\rangle$ iff (1) E is B-closed, relative to S, E, and (2) for all E' such that E' is B-closed, relative to S, E', we have $E' = E$ if $E' \subseteq E$.

A B-extension makes use only of B-defaults. The definitions, then, are equivalent to those of [12].

2.5 A PROBLEM

Of course, the fact that the logic BD satisfies the intuitive requirements of a number of examples is no guarantee that it is complete or even sound. In cases like this, it would be desirable to have a formal criterion of soundness and completeness that is intuitively satisfactory, and substantially different from the rather proof-theoretic formulations of Section 2. I do not think it will be easy to devise a semantics of this kind for BD, and believe that this is a side-effect of trying to formalize practical reasoning. I do not want to seem indifferent to the need for semantics; providing a useful model theoretic semantics for logics like BD is certainly an appropriate long-range goal. But for the time being, I will continue to rely on specific example-driven intuitions, and on the general logical intuitions that derive from the close relationship of the logic BD to a familiar formalism for nonmonotonic reasoning.

Pursuing this method reveals a residual problem with BD. Depending on how you look at it, it is either inadequate, or is highly incomplete as a system of practical reasoning.

Recall the train of thought that disclosed the need to prioritize defaults in the logic BD. (1) I decided at the outset to model practical reasoning by allowing both beliefs and desires to act as defaults. (2) Then I noticed that this model induced intuitively invalid cases of invalid "wishful thinking" in which desires were not properly constrained by beliefs. (3) To solve this problem, I prioritized B-defaults over D-defaults.

There is a good reason to allow desires to license default conclusions—this provides a natural way of modeling how goals are introduced into practical arguments. However, we learned that the way in which desires can enter into practical arguments has to be limited.

The following example shows that we will need to find further limitations of this kind.

Example 2.8. *The Reasoning in Natural Language.*

1. I'd like to have decaf coffee.
2. I can only have decaf coffee if decaf coffee is available.
3. *So:* Decaf coffee must be available.

Example 2.8, continued. *Formalizing the Premises.*

1. $\top \overset{D}{\hookrightarrow}$ `Have-Decaf`
2. `Have-Decaf` $\overset{B}{\hookrightarrow}$ `Decaf-Available`

This example yields just one BD-extension, the one that is generated by the following choices.

{`Have-Decaf`, `Decaf-Available`}

There are no prior beliefs in this example concerning the availability of decaf. The default desire $\top \overset{D}{\hookrightarrow}$ `Have-Decaf` is not conflicted, and so produces the conclusion `Have-Decaf`. Since I believe that `Decaf-Available` is a necessary condition for `Have-Decaf`, the additional conclusion is produced.

This reasoning is clearly a case of unsound, wishful thinking, but given what has been said so far, there is no evident criterion for separating the apparently sound reasoning of Examples 2.5–2.7 from the fallacious reasoning of Example 2.8.

You might conclude from this exercise that the system BD is unsound. I think it is more accurate to say that it is sound, as far as the examples that have been discussed here are concerned, and also sound as far as I know. *But*, without a mechanism for formalizing action and change, the system can only deliver an account of what extensions maximize desires, without contradicting any beliefs. There is nothing wrong with the conclusion reached in Example 2.8, as long as we think of it as representing an outcome that is maximally desirable, within the limits of what is believed. The fact that this is not very useful information—it does little good to know what outcomes are preferred, if we have no information about how to achieve these outcomes—simply shows that BD is expressively incomplete as a system of practical reasoning.

Planning formalisms of the sort that have been developed by AI-minded logicians—systems for reasoning about action and change—provide exactly what is needed to remedy this defect. In the next section, I will show how natural constraints that can be formulated in the extended logic BDP for belief, desire, and planning can eliminate extensions of the sort that BD produces in Example 2.8.

3 A FORMALISM FOR BELIEF, DESIRE, AND PLANNING

I will work with a simplified version of the *Situation Calculus*, [9, 7, 8].

3.1 REPRESENTING CHANGE AND PLANS IN BDP

BDP is an extension of BD incorporating a specialized first-order language containing an apparatus for

reasoning about actions and change; it will also use a modified extension definition that requires extensions to be grounded in plans.

The Situation Calculus uses a predicate **Holds** to denote a relation between fluents (i.e., dynamic properties) and situations; a functional constant **result** denotes a function from actions and situations to situations. Suppose that s_0 denotes an initial situation $\mathbf{s_0}$, that a_1 and a_2 denote actions $\mathbf{a_1}$ and $\mathbf{a_2}$, and that f denotes the fluent **f**. Then

(3.1) $\mathtt{Holds}(f, \mathtt{result}(a_2, \mathtt{result}(a_1, s_0)))$

is the standard way of expressing in the formalism that performing $\mathbf{a_1}$ and $\mathbf{a_2}$ in $\mathbf{s_0}$ will yield a situation in which **f** holds. There is no reasoning about goal selection in the Situation Calculus, and no way of distinguishing the fluents that are goals from the others. And there is no way of distinguishing adopted plans from the others; one can only say what goals a plan will achieve, in a given initial situation.

BDP is intended to formalize reasoning that results in the formation of plans. In choosing a BDP extension, an agent commits not only to beliefs and desires, but to a sequence of actions, i.e., a plan. This means that the Situation Calculus must be modified to provide for explicit commitment to a plan.

There are three ingredients to the modification. (1) BDP has a family $\mathtt{step}_1, \mathtt{step}_2, \ldots$ of individual constants; $\mathtt{step}_i$ denotes the ith step of the selected plan. (2) There is a sequence $\mathtt{Planlength}_i$, $i \in \omega$, of designated propositional constants; $\mathtt{Planlength}_i$ means that the designated plan has i steps. There is a special constant **null**; $\mathtt{step}_i = \mathtt{null}$ means that the ith step of the selected plan is undefined. (3) BDP has a family $\mathtt{Holds}_0$, $\mathtt{Holds}_1$ ... of 1-place predicates, rather than a single **Holds** relation. With this apparatus, there is no need to represent situations explicitly. The situation that results after the performance of i steps of the designated plan is represented implicitly by $\mathtt{Holds}_i$.

We will need to formulate desires about the future. For the simplified purposes of this paper I will introduce a predicate **Eventually**, satisfied by fluents which hold immediately after the selected plan is executed.

Actions are treated as individuals in BDP. There is a set of designated action constants and action functions. A *closed action term* is a term having the form t, where t either is an action constant, or is $f(t_1, \ldots, t_n)$, where $t_1, \ldots, t_n$ are terms containing no individual variables and f is an action function letter. AT is the set of closed action terms.

The workings of BDP can be illustrated with two sorts of examples: (1) blocks-world examples, and (2) more complicated formalizations of informal reasoning examples. The former examples are unrealistic, but their simplicity and familiarity makes it relatively easy to present detailed formalizations. This forum doesn't provide space to do justice to the latter examples; that task will have to be reserved for a more extended presentation of the ideas.

Take a blocks world with just two blocks, **a** and **b**, denoted respectively by **a** and **b**; there is a table, denoted by a constant **table**. Actions are denoted by terms of the form $\mathtt{move}(t_1, t_2)$. Suppose that in the initial situation **b** is on **a**. Then a theory selecting the simplest plan to get **a** on **b** will contain (among other formulas):

Example 3.1. *A blocks-world plan extension.*

$\mathtt{step}_1 = \mathtt{move}(\mathtt{b}, \mathtt{table})$
$\mathtt{step}_2 = \mathtt{move}(\mathtt{a}, \mathtt{b})$
$\mathtt{Planlength}_2$
$\mathtt{Holds}_0(\mathtt{on}(\mathtt{b}, \mathtt{a}))$, $\mathtt{Holds}_0(\mathtt{on}(\mathtt{a}, \mathtt{table}))$
$\mathtt{Holds}_1(\mathtt{on}(\mathtt{b}, \mathtt{table}))$, $\mathtt{Holds}_1(\mathtt{on}(\mathtt{a}, \mathtt{table}))$
$\mathtt{Holds}_2(\mathtt{on}(\mathtt{b}, \mathtt{table}))$, $\mathtt{Holds}_2(\mathtt{on}(\mathtt{a}, \mathtt{b}))$
$\neg\mathtt{Planlength}_3$
$\mathtt{step}_3 = \mathtt{null}$

3.2 FORMALIZING ACTION AND CHANGE IN BDP

It is straightforward to translate the usual policies for formalizing action and change in the Situation Calculus to BDP. The details are omitted in this presentation; for simplicity, I assume that frame axioms are monotonic.

3.3 INFORMAL DESCRIPTION OF BDP EXTENSIONS

As in BD, a basis will consist not only of monotonic axioms, but of belief and desire defaults. As before, belief defaults have the form $A \overset{\mathrm{B}}{\hookrightarrow} B$. Any desire default of the form $A \overset{\mathrm{D}}{\hookrightarrow} B$ is allowed, but in realistic cases we are interested in desires that are future-directed. In the simplified context of the present paper, that means desires of the form $A \overset{\mathrm{D}}{\hookrightarrow} \mathtt{Eventually}(f)$.

We wish extensions to choose *feasible* options—ones that are not only desirable and compatible with beliefs, but that can be secured by acting on a plan. We ensure this by first constructing BD extensions. Assuming that the beliefs and desires have to do only with world states (i.e., with whether or not fluents

hold), and not with plan length or selection of actions, a BD extension will present a picture of things as the planning agent would like them to be, but without any specific information about actions that would realize this desirable picture.

I do not exclude *prima facie* beliefs such as

$$\top \overset{B}{\hookrightarrow} \texttt{Holds}_3(f),$$

or *prima facie* desires such as

$$\top \overset{D}{\hookrightarrow} \texttt{Planlength}_5$$

or

$$\top \overset{D}{\hookrightarrow} \texttt{step}_1 = \texttt{move(a, table)}.$$

In the simplest case, though, all belief defaults have conclusions that are literals of the form $\texttt{Holds}_0(f)$ or $\neg\texttt{Holds}_0(f)$, and all desire defaults have conclusions that are literals of the form $\texttt{Eventually}(f)$ or $\neg\texttt{Eventually}(f)$. In this case (assuming that the monotonic theory M provides only general domain information) a BD extension will provide initial conditions and goals—precisely the sort of input that a classical planning algorithm would need.

A BDP plan-description over a set of closed action terms AT is a set of formulas consisting of a choice of a plan length and a selection of plan steps up to the plan length. For any BDP plan-description PD, then, there will be an n and a set of terms $\{t_1, \ldots, t_n\} \subseteq AT$, such that:

$$PD = \{\texttt{Planlength}_n, \texttt{step}_1 = t_1, \ldots, \texttt{step}_n = t_n\}.$$

A BDP proto-plan over a BDP-basis $\mathcal{S}$, where $\mathcal{S} = \langle M, NB, ND, AT\rangle$ is a consistent set of the form $\text{Th}_{\text{FOL}}(E_1 \cup PD)$, where E_1 is a BD extension of $\mathcal{S}$ and PD is a plan-description over PD. A solved planning problem will involve a *means*, a series of actions calculated to achieve certain goals; and, of course, it will involve the goals or ends themselves. BDP proto-plans determine a means, in the form of a plan-description; they determine ends, in the form of the desires activated in a BD-extension.

But a proto-plan needn't correspond to a feasible plan; it may involve wishful thinking. There is nothing to ensure that the means of a proto-plan cause the ends; a goal may belong to a proto-plan simply because it is the conclusion of a default desire in ND.

Therefore, we filter out the proto-plans that involve wishful thinking. A BDP extension of a BD-basis $\mathcal{S} = \langle M, NB, ND\rangle$ is a BDP proto-plan E over $\mathcal{S}$ with plan-description PD such that E is a B extension of $\langle M \cup PD, NB\rangle$. Thus, every formula in E—including the formulas that represent desires or ends—follows merely from beliefs, given the steps of its plan-description.

The technical definitions are given below, in Section 3.5; before turning to these, I will illustrate the ideas with examples.

3.4 EXAMPLES OF BDP REASONING AND EXTENSIONS

Blocks-world examples

All blocks-world examples use the obvious blocks-world axioms (which are not presented here). In the first example, the agent has unconflicted *prima facie* beliefs about the initial conditions, and an unconflicted *prima facie* desire to have block **a** on block **b**. In BDP, this case works like a traditional planning problem in which the default beliefs give the initial conditions, and the desire gives the goal.

Example 3.2. *Simple, unconflicted beliefs and desires.*
Defaults:

$$NB = \{\top \overset{B}{\hookrightarrow} \texttt{Holds}_0(\texttt{on(a, table)}),\ \top \overset{B}{\hookrightarrow} \texttt{Holds}_0(\texttt{on(b, table)}),\ \top \overset{B}{\hookrightarrow} \texttt{Holds}_0(\texttt{on(c, table)})\}$$

$$ND = \{\top \overset{D}{\hookrightarrow} \texttt{Eventually(on(a, b))}\}$$

BDP extensions.

There are in fact many BDP extensions, corresponding to the various sequences of moves that will get **a** on **b**. The shortest such plan is generated by the following formulas:

$$(Pl_1): \quad \{\texttt{Planlength}_1, \texttt{step}_1 = \texttt{move(a, b)}\}$$

Discussion.
Why exactly does (Pl_1) generate a BDP plan? First, note that

$$(BD\text{-}Ext_1): \quad \{\texttt{Holds}_0(\texttt{on(a, table)}), \texttt{Holds}_0(\texttt{on(b, table)}), \texttt{Holds}_0(\texttt{on(c, table)}), \texttt{Eventually(on(a, b))}\}$$

generates a BD-extension in this example. (In fact, this is the only BD-extension.)

Second, note that Pl_1 is a BDP plan-description. Since $BD\text{-}Ext_1 \cup Pl_1$ is consistent, its logical closure, PD_1, is

a BDP proto-plan. It is easy to check that PD_1 is a B-extension of the information given in the example; all that is involved is verifying that $\texttt{Holds}_1(\text{on}(a,b))$ follows from the initial conditions and the appropriate causal axiom for **move**. Therefore, PD_1 is a BDP extension.

In the next example, the agent has the same unconflicted *prima facie* beliefs about the initial conditions, and conflicted *prima facie* desires: a desire to have block **a** on block **c** and a desire to have block **b** on block **c**. This yields two sorts of BDP extensions: those corresponding to plans for putting **a** on **c** and those corresponding to plans for putting **b** on **c**. Since these goals are incompatible, there are no BDP extensions in which both goals are satisfied.

Example 3.3. *Conflicted desires, unconflicted beliefs. Defaults:*
These are like Example 3.2, except that:

$$ND = \{\top \overset{D}{\hookrightarrow} \texttt{Eventually(on(a, c))},\ \top \overset{D}{\hookrightarrow} \texttt{Eventually(on(b, c))}\}$$

BDP Extensions.
The simplest BDP extensions (corresponding to the shortest plans) are generated by the following choices:

*BDP-Ext*3.3$_1$:
$\{\texttt{Planlength}_1, \texttt{step}_1 = \texttt{move(a, c)}\}$
*BDP-Ext*3.3$_2$:
$\{\texttt{Planlength}_1, \texttt{step}_1 = \texttt{move(b, c)}\}$

In this sort of case, the best way to choose among the alternative BDP extensions would be to use plan evaluation methods that somehow combine criteria having to do with the simplicity of the plan and with the utilities of the outcomes.

In the next example, the agent has contradictory *prima facie* beliefs about the initial conditions: that block **a** is on block **c** and that block **b** is on block **c**. There is one *prima facie* desire: to have **c** clear. Here, there are two sorts of extensions: those corresponding to plans for getting **a** on the table, assuming that **a** is on **b**, and those corresponding to plans for getting **b** on the table, assuming that **b** is on **a**.

Example 3.4. *Unconflicted desires, conflicted beliefs. Defaults:*

$$NB = \{\top \overset{B}{\hookrightarrow} \texttt{Holds}_0\texttt{(on(a, c))},\ \top \overset{B}{\hookrightarrow} \texttt{Holds}_0\texttt{(on(b, c))},\ \top \overset{B}{\hookrightarrow} \neg\texttt{Holds}_0\texttt{(on(a, b))},\ \top \overset{B}{\hookrightarrow} \neg\texttt{Holds}_0\texttt{(on(b, a))},\ \top \overset{B}{\hookrightarrow} \texttt{Holds}_0\texttt{(on(c, table))}\}$$
$$ND = \{\top \overset{D}{\hookrightarrow} \texttt{Eventually(Clear(c))}\}$$

BDP extensions.
The simplest extensions (corresponding to the shortest plans) are generated by the following choices:

*BDP-Ext*3.4$_1$:
$\{\texttt{Planlength}_1, \texttt{step}_1 = \texttt{move(a, table)}\}$
*BDP-Ext*3.4$_2$:
$\{\texttt{Planlength}_1, \texttt{step}_1 = \texttt{move(b, table)}\}$

Remember, this is a case in which the agent has a reason to believe that **a** is on **c** and a reason to believe that **b** is on **c**. It would not be at all appropriate to choose between the two extensions according to utilities, since the plan with the higher utility could rest on a false assumption and be infeasible.[4]

Example 3.5. *Conflicted desires, conflicted beliefs. Defaults: These are like Example 3.4, except that:*

$$\text{ND} = \{\top \overset{D}{\hookrightarrow} \texttt{Eventually(on(a, b))},\ \top \overset{D}{\hookrightarrow} \texttt{Eventually(on(b, a))}\}$$

BDP extensions. The simplest extensions are generated by the following choices:

*BDP-Ext*3.5$_1$:
$\{\texttt{Holds}_0\texttt{(on(a, c))}, \texttt{Planlength}_1, \texttt{step}_1 = \texttt{move(a, b)}\}$
*BDP-Ext*3.5$_2$:
$\{\texttt{Holds}_0\texttt{(on(a, c))}, \texttt{Planlength}_1, \texttt{step}_1 = \texttt{move(b, a)}\}$
*BDP-Ext*3.5$_3$:
$\{\texttt{Holds}_0\texttt{(on(b, c))}, \texttt{Planlength}_1, \texttt{step}_1 = \texttt{move(a, b)}\}$
*BDP-Ext*3.5$_4$:
$\{\texttt{Holds}_0\texttt{(on(b, c))}, \texttt{Planlength}_1, \texttt{step}_1 = \texttt{move(b, a)}\}$

In this case, the extensions are a sort of cross-product of the alternatives presented by the conflicting beliefs

[4] This sort of conflicting default has nothing in particular to do with desires. Such cases could occur, in principle, in traditional nonmonotonic planning formalisms. But they tend to be ignored, since these formalisms tend to confine nonmonotonicity to frame axioms and to treat multiple extensions as anomalies.

and those presented by the conflicting desires. In the absence of any opportunity to gather information, the best way to attack this would be to first choose the most plausible or likely beliefs, and then to choose the plans that, within these, BDP extensions, maximize utility.

A more complex reasoning example

There isn't space in this presentation of the ideas to explore a variety of examples. I will only give one more example to indicate that BDP could deliver a reasonable account of informal practical argumentation.

Example 3.6. *A case with two extensions.*
The informal reasoning:

This is the same as the reasoning in Example 2.7.

Formalization in BDP:

1. There are four fluents: `rain`, `wet`, `home`, and `too-far`. The last fluent is introduced to make continuing to hike incompatible with going home later.
2. The monotonic axioms include $\neg\mathtt{Holds}_i(\mathtt{too\text{-}far})$ for $1 \leq i \leq 3$, as well as appropriate axioms for the actions.
3. $NB = \{\top \overset{B}{\hookrightarrow} \mathtt{Holds}_2(\mathtt{rain}),$
   $\mathtt{Holds}_2(\mathtt{rain}) \overset{B}{\hookrightarrow} \mathtt{Holds}_3(\mathtt{wet}),$
   $\mathtt{Holds}_2(\mathtt{rain}) \overset{B}{\hookrightarrow}$
   $\neg[\mathtt{Eventually}(\mathtt{wet}) \leftrightarrow \mathtt{Eventually}(\mathtt{home})]\}$
4. $ND = \{\top \overset{D}{\hookrightarrow} \neg\mathtt{Holds}_3(\mathtt{wet}),$
   $\top \overset{D}{\hookrightarrow} \neg\mathtt{Eventually}(\mathtt{wet}),$
   $\top \overset{D}{\hookrightarrow} \neg\mathtt{Eventually}(\mathtt{home})\}$
5. There are three actions:
   (a) `wait` has no preconditions and no effects.
   (b) `walk-home` has $\neg$`too-far` as its only precondition. It has `home` as its only positive effect and `wet` as its only negative effect.
   (c) `hike-on` has no preconditions and has `too-far` as its only positive effect.

Here, there are two kinds of BDP extensions: the ones that satisfy the desire to not be wet by performing `walk-home`, but frustrate the desire to be away from home, and the ones that perform `hike-on` and so satisfy the desire to be away from home but frustrate the desire to be dry.

3.5 FORMAL DESCRIPTION OF BDP EXTENSIONS

Definition 3.10. *BDP-basis*
A BDP-basis is a package $\mathcal{S} = \langle M, NB, ND, AT\rangle$, where $\langle M, NB, ND\rangle$ is a BD-basis and AT is a set of action terms.

Definition 3.11. *BDP plan-description.*
PD is a BDP plan-description of length n over AT if for some $t_1, \ldots, t_n \in AT$:

(1) $\mathtt{Planlength}_n \in PD$,
(2) $t_1, \ldots, t_n \subseteq PD$.

Definition 3.12. *BDP proto-plan.*
T is a BDP proto-plan over a BDP-basis $\langle M, NB, ND, AT\rangle$ iff for some BD extension E_1 of $\langle M, NB, ND\rangle$ and plan-description PD over AT, $T = E_1 \cup PD$

Definition 3.13. *BDP extension.*
E is a BDP extension of a BDP basis $\mathcal{S} = \langle M, NB, ND, AT\rangle$ iff (1) E is a proto-plan over $\mathcal{S}$ with plan-description PD, and (2) E is a B extension of $\langle M \cup PD, NB\rangle$.

3.6 INTERACTING DEFAULTS

In applications of nonmonotonic logic that are at all complex, one has to be on the lookout for interactions between defaults that may call for prioritization. In this case, I have not come up with any obvious examples of that sort, even in more complicated domains where the nonmonotonic apparatus is used to solve the frame problem.

3.7 EXTENSION EVALUATION

BDP leaves conflicts between desires unresolved if they are not removed by feasibility considerations. This is illustrated by the two extensions generated in Example 3.6; the reasoning provided by BDP does not recommend either solution to the hiker's dilemma in Example 2.7. Although it does focus the practical problem by providing two alternatives, it provides no mechanism for choosing between (i) remaining wet and (ii) getting dry but aborting the hike. To resolve dilemmas like this, we will have to make a direct comparison between the costs of the two alternatives.

In the hiking example, this means that—if we wish to resolve the problem generally—we will need to find some way to balance the discomfort and inconvenience

of wearing wet clothes against the frustration of aborting the hike. The appropriate analytical methods for such problems are those of multiattribute utility theory (see, for instance, [5]). These methods work best on cases where there are recurrent, similar decision problems, where there are appropriate scales for measuring the attributes, and where the tradeoffs under consideration are relatively context-independent. This means that analytical methods are unlikely to yield a satisfactory resolution of the hiker's dilemma. Nevertheless, utility analyses can be usefully applied to a broad range of phenomena, and the combination of qualitative reasoning of the sort provided by BDP seems to be a potentially useful one—BDP can eliminate some alternatives as excluded incompatible with feasible maximization of desires, and perhaps in some cases it could deliver a limited range of alternatives appropriate for utility analysis. It remains to be seen, of course, whether this sort of reasoning can be carried out efficiently in cases of practical interest.

In all of the cases considered so far, the element of risk is negligable. BDP does not rely on probabilistic reasoning; its nonmonotonic approach to belief simply excludes alternatives from practical consideration that are incompatible with beliefs. This approach is appropriate in cases where it is important to focus the reasoning, and the consequences of acting inappropriately on false assumptions are bearable. I haven't begun to think through the details of how to extend BDP to cases where risk needs to be reasoned about explicitly; I am sure, however, that this extension would have to involve an integration of some sort of probabilistic and default reasoning; it would certainly require a major reworking of the treatment of belief, and for the moment I think it is best to concentrate on the case in which risk can be neglected.

4 AGENT ARCHITECTURES

Ideas from the AI planning paradigm, together with the popularity of agent design, have inspired the development of *BDI architectures*: frameworks for designing agents that take the attitudes of belief, desire, and intention to be central.[5]

Here is a brief list of considerations that make it desirable to integrate core functions into the reasoning processes on which an agent architecture is based.

1. Belief revision only becomes nontrivial when the reasoning that produces beliefs is nonmonotonic; typically, beliefs that need to be discarded in the course of belief revision will be ones that are created by fallible reasoning processes, and a belief revision policy that fails to take these processes into account will neglect crucial information.[6] This suggests that nonmonotonic reasoning should be an important component of planning. However, although nonmonotonic logics are often integrated into inertial constraints on state dynamics in planning formalisms (i.e., they are often used to address the frame problem), nonmonotonic domain reasoning is generally ignored in planning formalisms. The BDP formalism takes nonmonotonic domain reasoning into account explicitly.

2. For replanning in light of revised beliefs, it could be useful to record the dependencies of plans on beliefs.

3. These plan dependencies could also be invoked in qualitative assessments of risk. A plan can be tested for riskiness by comparing the value of the expected outcome with the value of the outcomes that ensue if various fallible beliefs on which the plan depends fail. These comparisons would be facilitated by a nonmonotonic logic of belief that provided some qualitative assessment of the reliability of beliefs.

4. In a further development of this line of thought, risk assessment could be integrated into planning by seeking to avoid risky plans.

5. Goals come from desires, and—as I pointed out and illustrated in Section 2, desires can be mutually inconsistent in light of beliefs. This provides strong motives for the commonsense planning strategy of planning with flexible goals, and discarding desires that prove to be too costly in light of feasibility considerations. (The vacation planning example from Section 1 was intended to illustrate this point.)

6. A similar point applies to intentions. Agents may need to cut their losses. It is unreasonable to cling to an intention when it becomes too costly to do so, even though the intention remains feasible in principle. By carrying out the planning process in such a way that dependencies on fallible beliefs are recorded, as well as correlations of these beliefs to cost factors, it may be possible to provide in advance for a flexible readjustment of intentions.

[5] See [3, 11, 10]. The ideas behind BDI architectures also have a philosophical dimension; see [2].

[6] Much of the literature on belief revision ignores the relationship to nonmonotonic domain reasoning, so this claim has to be regarded as controversial.

The logic BDI suggests an agent made up of attitudes that, after all, are not such a radical departure from the BDI model; you could call it a B^2D^2I agent. Intentions, as before, are commitments to adopted plans and their components. Beliefs and desires are divided into two varieties: *prima facie* and all-things-considered. Practical reasoning, which is a generalization of AI-style planning, consists in converting *prima-facie* into all-things-considered attitudes, and in providing a means in terms of feasible actions of satisfying the all-things-considered desires.

A B^2D^2I agent is in a much better position than a BDI agent to replan in light of adverse experience, because it has alternative clusters of goals to fall back on. This sort of flexibility is called for in philosophical studies of practical reasoning such as [2], but was never fully realized in BD architectures.

References

[1] David E. Bell, Ralph L. Keeney, and Howard Raiffa, editors. *Conflicting Objectives in Decisions.* John Wiley and Sons, New York, 1977.

[2] Michael Bratman. *Intentions, Plans and Practical Reason.* Harvard University Press, 1987.

[3] Michael Bratman, David Israel, and Martha Pollack. Plans and resource bounded practical reasoning. *Computational Intelligence*, 4:349–355, 1988.

[4] Gerhardt Brewka. Reasoning about priorities in default logic. In Barbara Hayes-Roth and Richard Korf, editors, *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 940–945, Menlo Park, California, 1994. American Association for Artificial Intelligence, AAAI Press.

[5] Peter C. Fishburn. Independent preferences. In John Eatwell, Murray Milgate, and Peter Newman, editors, *The New Palgrave: Utility and Probability*, pages 121–127. Macmillan, New York, 1987.

[6] John F. Horty, Richmond Thomason, and David Touretzky. A skeptical theory of inheritance in nonmonotonic semantic networks. *Artificial Intelligence*, 42:311–349, 1990.

[7] Vladimir Lifschitz. Formal theories of action. In Frank M. Brown, editor, *Proceedings of the 1987 Workshop on the Frame Problem*, pages 35–57, Los Altos, California, 1987. Morgan Kaufmann.

[8] Vladimir Lifschitz. Towards a metatheory of action. In James Allen, Richard Fikes, and Erik Sandewall, editors, *KR'91: Principles of Knowledge Representation and Reasoning*, pages 376–386. Morgan Kaufmann, San Mateo, California, 1991.

[9] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

[10] Anand S. Rao and Michael Georgeff. An abstract architecture for rational agents. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *KR'92. Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*, pages 439–449. Morgan Kaufmann, San Mateo, California, 1992.

[11] Anand S. Rao and Michael P. Georgeff. Modeling rational agents within a BDI-architecture. In James F. Allen, Richard Fikes, and Erik Sandewall, editors, *KR'91: Principles of Knowledge Representation and Reasoning*, pages 473–484. Morgan Kaufmann, San Mateo, California, 1991.

[12] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–32, 1980.

[13] Richmond H. Thomason. Towards a logical theory of practical reasoning. In *Working Notes of the AAAI Spring Symposium on Reasoning about Mental States*, pages 133–142, Menlo Park, CA, 1993. American Association for Artificial Intelligence.

[14] Frank Veltman. Defaults in update semantics. *Journal of Philosophical Logic*, 25(2):221–261, 1996.

Panel Abstracts

Practical Knowledge Representation and the DARPA High Performance Knowledge Bases Project

Adam Pease
Teknowledge
1810 Embarcadero
Palo Alto, CA 94303
USA
apease@teknowledge.com

Vinay Chaudhri
SRI International
333 Ravenswood Ave
Menlo Park, CA 94025
USA
chaudhri@ai.sri.com

Fritz Lehmann
Cycorp
3721 Executive Cntr Dr
Austin, TX 78731
USA
fritz@cyc.com

Adam Farquhar *
Schlumberger
8311 North FM 620 Road,
Austin TX 78726
USA
afarquhar@slb.com

Abstract

We address the experiences of the DARPA High Performance Knowledge Bases (HPKB) (Cohen et al., 1998) project in practical knowledge representation. The purpose of the HPKB project was to develop new techniques for rapid development of knowledge bases. The goal of this paper is to describe several technical issues that arose in creation of practical KB content.

1. HPKB PROJECT

1.1. EXPERIMENTS

The project had two main objectives: first, to advance the science of Artificial Intelligence Knowledge Representation and Knowledge Base content creation, and second, to apply these technologies to create applications with utility to the Department of Defense. The applications were specified as two Challenge Problems (CPs). The first was the Crisis Management CP, an effort to develop an automated question answering system that met the needs of analysts who must be informed about emerging world crises. The second was the Battlespace Challenge Problem. This effort covered two knowledge-based systems. One reasoned about battlefield engineering tasks such as workaround computation; the other critiqued battle plans. This paper addresses issues primarily from the experiences of the Crisis Management CP.

* The author performed this work while a member of the Knowledge Systems Laboratory, Stanford University

1.2. PROJECT ORGANIZATION

Two teams worked on these challenge problems. In the Crisis Management CP, one team used Cyc (Lenat, 1995) and its MELD (Cycorp, 1997) representation language. Another used KIF (Genesereth & Fikes, 1992) and the SNARK (Stickel et al., 1994) and ATP theorem provers.

HPKB was a very large project and many aspects are not even mentioned in this paper. The interested reader should refer to the HPKB web site (HPKB Web, 1999) and publications list (HPKB Pubs, 1999).

2. TRADEOFFS IN THEORY CREATION

There is a cost in creating reusable representations. It is more costly to create representations that will be reusable across multiple domains than it is to create a representation that is suitable for just one application.

We believe there is a need for a more formal development process that is built on some of the best practices from the software engineering community. It is always easier to create specific and limited content as opposed to crafting a general domain theory. The challenge is to build time into the development process for planning and systems analysis, design, implementation, testing, and *rework and generalization.* Much like the spiral development model advocated by Booch (Booch, 1994) and others, a good development process iterates through these stages several times during a development process. One possible instantiation of this process would be as follows:

2.1. DEVELOPMENT PROCESS

Planning and systems analysis. It is essential to determine the need that the knowledge must fulfill. Will it be used for inference? To define a semantics for natural language interpretation? As an interlingua for

cooperating agents or software modules? Each of these applications will entail a different emphasis on the richness of the formalization.

Also considered should be the performance requirements of the implementation. How fast should the resulting inference be? Will the knowledge base need to be augmented with a significant amount of instance data? Is logical completeness a necessity? Answering these questions will help to determine how expressive the knowledge representation can be, which will in turn partially determine the inference engine that needs to be employed.

We should note that in the HPKB project, a great deal of the systems analysis phase was done for the knowledge base developers by providing them with a Challenge Problem (Schrag, 1999:2) that specified and detailed the scope and purpose of the experiments that were to come. A great deal of informally specified knowledge was also provided.

Design. One way to design a knowledge base is initially to specify it informally. The engineer creates English examples illustrating sample reasoning chains. Glossaries with English definitions are created. It can also be useful to create a taxonomy as a skeleton on which the theory can be developed.

Implementation. As in software development, if the two previous phases are done properly, the implementation phase can proceed quickly. It is important that all members of the development team participate in the first two phases. Also helpful is a formal review process led by a chief knowledge *architect.*

Knowledge architects, software architects, and building architects all have similar roles. While they do not control every detail of a project, they set the overall design, standards, and aesthetics. A knowledge architect provides guidance to his team about how to meet project requirements, find a balance in tradeoffs between development speed and implementation generality, maintain consistent approaches across diverse team members, and set standards for reviews and documentation. A good architect manages by objectives and standards, which result in an implementation that speaks with one voice while allowing participants the freedom to innovate.

Testing. While this phase is obvious for any knowledge base that is to be used in a computational system, performing systematic testing is often ignored. If the knowledge base has been developed in a modular manner, an equivalent to *unit testing* can be performed on each small theory. Unit testing allows for testing of greater coverage than final *integration testing.*

Rework and Generalization. This phase is the most often ignored simply because of the dynamics of most research projects. Once the practical objectives of the sponsors have been achieved, little time or money remains in the project to correct shortcuts that may have been made. However, this phase is possibly the most important if incremental scientific results are to be achieved.

Any large scale project will necessarily go through the above phases several times. A good knowledge engineering process has many similarities to a good software engineering process.

3. THEORY REUSE

Both teams reused the HPKB upper level (HPKB-UL) ontology, derived from Cyc, during the project. The representation for the temporal knowledge available in the HPKB upper ontology was very well designed. From the HPKB-UL, we also used representation for communicative actions, slots on actions (agent roles), and the primitives for representing paths. For one team, reusing these theories required translating the representation, extracting portions of the input ontology for use, and doing limited reformulation. There was also the need to further extend the library of the representation primitives for causality, scales, actions, processes, and qualitative influences.

The Cyc-based team had access to the entire Cyc knowledge base. In addition to areas mentioned for the upper level, there are good theories for concrete physical domains of all sorts. Theories of belief, goals, trust, and the expression of causality in nondeterministic human events are essential and less well developed.

HPKB had a good record of reusing terms and basic statements about terms. Developers gained a great deal of value from inheriting a large set of precise distinctions about things in the world, such as the differences among a goal, a plan, and a desire. However, comparatively little reuse of general rules was evident. This can be explained in several ways:

- It's hard to write truly general rules.
- Insufficient effort has been placed into writing general rules because of the pressures of day-to-day results.
- Practicalities of inference are such that a long chain of reasoning involving general rules doesn't work in a reasonable amount of time. One has to "short-circuit" the deep reasoning with special-purpose rules that make the inference tractable.

As an example of reuse, consider the following inference task performed by our system:

> *What risks can Iran expect in sponsoring a terrorist attack in Saudi Arabia?*

To answer questions of this type, one team developed a simple cause-effect model. All the predicates below, including *cause-event-event*, *beneficiary*, and *maleficiary* were reused from the HPKB-UL. Even though we capture only direct effects of an action, this simple model was effective in practice. This example illustrates the reuse of notions of causality that were already conceptualized in the HPKB-UL. The following is an example application of these representation primitives.

```
(forall ((?terrorist-attack
          terrorist-attack)
         (?agent agent))
(=>
  (performed-by ?terrorist-attack ?agent)
  (exists
     ((?punishment punishment))
     (and
        (causes-event-event
           ?terrorist-attack
           ?punishment)
        (maleficiary ?punishment ?agent)
        (object-acted-on
           ?punishment ?agent)))))

(forall ((?action action)
         (?action1 action1))
 (implies
   (and
      (causes-event-event ?action ?action1)
      (performed-by ?action ?agent)
      (beneficiary ?action1 ?agent))
   (benefit-of-action
      ?action ?action1 ?agent)))
```

A detailed description of technical problems encountered in reuse is available in (Cohen et al., 1999) (Chaudhri et al., 2000). Even though we reused representations for actions and casuality from HPKB-UL, significant additional representation work needed to be done. This suggests that a representation library for actions, causality, and qualitative influences needs to be extended. The theoretical KR community is invited to study the HPKB-UL and propose representational modules to be included in it.

4. PRACTICAL REPRESENTATIONAL ISSUES

There was a lack of principles for designing taxonomies. As a result, creating and maintaining a taxonomy of primitive concepts became increasingly difficult as its size grew. Conventional description logic techniques do not help in creating taxonomies that contain a large number of primitive concepts. Better principles for taxonomy design are needed.

There was also the need to "hand-compile" deep reasoning out into special-purpose theories that had tractable inference chains.

4.1. TAXONOMY

Like many other KBs, the class-subclass taxonomy was an overarching organizing principle in our HPKB KB. A *class-subclass* taxonomy serves as an indexing aid to find knowledge and add new knowledge, and to serve as a method to efficiently write axioms by using inheritance.

While designing the taxonomies for the HPKB project, we encountered the following problems:

1. As the taxonomy got bigger, it became increasingly difficult to add new concepts to it. As a result, there were concepts that had incorrect positions in the taxonomy:

- Some concepts had missing links. A class has a missing super-class link if it is a subclass of another class B, but the subclass relationship is not declared.
- Some concepts had wrong links. A class has a wrong link in a taxonomy if it is a direct subclass of B, but the subclass relationship does not hold true.

2. We were encountering concepts that were being created by a cross-product of two sets of concepts, for example:

```
{International, transnational, subnational,
national} x {organization, agent}
{support, oppose} x {attack, terrorist-
attack, chemical-attack}    {Humanitarian,
political, military, diplomatic} x
{Organization, Action}
```

3. Some concepts had a very large number of subclasses. In some cases, this was due to orthogonal ways to categorize a concept. As a result, such categorizations were not mutually disjoint. Large fan-outs made it cumbersome to navigate through the taxonomy. As an example, consider the following snippet from the taxonomy representing organizations.

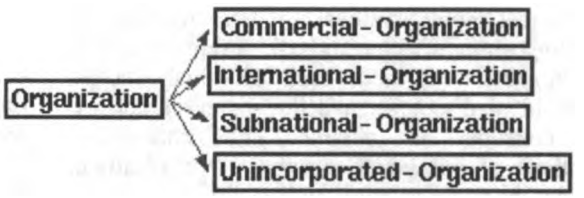


Figure 1. A portion of a taxonomy representing organizations showing orthogonal categorizations

While the categorization of commercial organization and unincorporated organization is based on the legal

status of an organization, the categorization of international organization and subnational organization is based on extent of operations. Mixing such orthogonal categorizations adds to the complexity of the taxonomy.

4. If two classes are disjoint, the disjointness relationship must be declared.

5. There should be no redundant classes representing identical concepts.

A taxonomy is well designed if it is free from all the problems mentioned above. Ensuring these properties in a small taxonomy is easy even if it is done manually. However, as the taxonomy size grows, making taxonomy well structured manually is very time consuming. These problems are indicative of a poor design methodology for developing taxonomies. We argue below that these problems go away if one takes a more principled approach to developing these taxonomies and supports additional constructs to structure the taxonomies.

If every concept has necessary and sufficient definitions, one can use a classifier to help alleviate Problem 1. In practice, we found that too many concepts were primitive and did not have necessary and sufficient definitions. Therefore, we cannot use a classifier. Problem 1 stems from the fact that the taxonomy itself is getting too complex. For example, a concept is linked or needs to be linked to too many different places. As a result, defining a new primitive concept involves manually encoding its relationship to numerous other primitive concepts -- a process that is error prone. One would hope that the process of organizing such concepts into a taxonomy would be considerably simpler than doing the same thing for the original concepts.

We need principles for taxonomy design that can enable us to economically create and maintain large taxonomies of primitive concepts.

4.2. COMPOSABLE REPRESENTATIONS

We believe that representations are more *reusable* if they are *compositionally* constructed. A representation is compositional if it represents each individual concept in the domain of discourse, and the representation of complex concepts is obtained by composing representations of individual concepts. To illustrate this, consider the representation of the following:

```
The USA conducts a peacekeeping mission.
```

In this example, we can use several different representations. One degenerate representation might be

```
UsConductOfPeacekeeping
```

This representation compiles all the semantic features of the English statement into a symbol. A more reasonable representation might be

```
(and
  (instance-of ?Y PeacekeepingOperation)
  (performedBy ?X ?Y)
  (members ?X USMilitaryOrganization))
```

in which the action has been expanded to describe an action type and detail about the performer of the action. We can further decompose the action by describing it as an event that has the purpose of maintaining a particular state.

```
(and
  (toMaintain ?Y PeaceAccord)
  (instance-of ?Y MilitaryOperation)
  (performedBy ?X ?Y)
  (members ?X USMilitaryOrganization))
```

(Schrag, 1999:1) has proposed the following compositionality hypothesis: noncompositional representations are inexpensive to build but they are brittle with respect to weak problem generalizations and must be re-engineered (for example, into compositional representations) or replaced.

According to the compositionality hypothesis, the first representation is inferior to the later versions. However, although many knowledge engineers would have a strong intuition that the later representations are superior, there is no strong empirical basis for the proposed criticism of the first representation. One approach that would admit the first representation as acceptable would be to add additional terms to the KB and give a more complete definition to it. Thus, even if the first representation is noncompositional, it is amenable to generalization if an application requires it.

The relative comparison between the two representations is unlikely to have a context-independent answer. If in the current application we never need to represent or reason with *conduct*, *mission*, or *peacekeeping*, other than talking about "conduct peacekeeping mission", the less expressive representation is adequate. One can certainly argue that the first representation is less reusable. However, that depends on the next application. If we use the first representation, and the next application requires us to represent or reason with *conduct*, *mission*, or *peacekeeping*, it is possible to add them to the KB and use them to define `UsConductOfPeacekeeping`. This may be studied more formally with an analytical model as follows.

Suppose we design two representations, one of which uses *n1* terms and the other uses *n2* terms. Suppose cost/term is *c* and is constant in both cases. The cost for building a KB for the two cases is $c*n1$ and $c*n2$, respectively.

If speeding up KB construction time for just one application is the objective, a compositional representation can be bad! However, if we also care about reuse, that may not be necessarily so. Does compositionality enable reuse? We cannot find out until we run replicated trials.

Suppose we reuse the KB for a new application. This new application requires the same knowledge fragment that we have already coded but requires a different compositionality, and we end up defining *n3* new terms for the first representation and *n4* new terms for the second representation. It is possible that either of *n3* or *n4* is zero. The cost for the new application is *c*n3* and *c*n4*, respectively.

The objective should be to minimize *c*(n1+n3)* or *c*(n2+n4)*. The model can be generalized to N applications. The parameter *c* can be viewed as time to construct a KB, and thus linked directly to the program goal of speeding up the KB construction time. Further, this model allows us to do the following:

a) Measure whether it is really worth decomposing a representation

b) Amortize the higher cost of decomposition over a number of applications

c) Make explicit the relationship between reuse and compositionality

Exploring this tradeoff is open for future work.

4.3. "COMPILED" REPRESENTATIONS

One of the HPKB Challenge Problems dealt with reasoning about economic actions. One might encode the following chain:

```
There exist economic actions
which open markets -
    opening markets encourages
    exports -
        increasing exports improves
        a country's trade balance -
            positive trade balance
            improves economic health -
                all countries are
                interested in
                economic health
```

However, it may be that in practice, because of the complexity and compositionality of each of the encoded statements, and the depth of the inference, such a reasoning chain does not terminate in a reasonable amount of time. While an inference of depth five may not seem very taxing, consider the fact that this set of rules exists in a very large KB along with tens of thousands of others. The task of matching these particular rules and determining that huge numbers of others are irrelevant is time consuming.

The result is that to create a reasoning system that reaches a conclusion in a short amount of time, one might have to encode

```
There is a set of actions
which open markets -
    opening markets contributes
    to economic health -
        all countries are interested
        in economic health
```

along with defining a set of actions as subclasses of "opening markets" actions.

The goals of a project can strongly bias a knowledge engineer to the second representation. If a research team is scored, or a development team is paid on the basis of "correct" answers, compositionality and deep reasoning will be sacrificed.

4.4. METRICS

For any practical KB content creation work, there is a need to state crisply the competence level of a KB, and to make claims about increasing competence as the time goes along. Even though we know that there is an intuitive relationship between the size of a KB and its competence, there is no foolproof way functionally to relate the size to competence. As an approximate measure, we used the axiom count in a KB as one measure of competence.

An early challenge during the project was to define what counts as an axiom. Given that there is no universal way to count axioms, and that the axiom counts are sensitive to the modeling style and the language, we developed the following scheme for categorization of axioms in a KB.

- **Constants** are any names in the KB, whether an individual, class, relation, function, or a KB module
- **Structural statements** are ground statements using any of (Cyc term/Ontolingua term) `##isa/instance-of`, `##genls/subclass-of`, `##genlPreds/subrelation-of`, `##disjointWith/disjoint`, `##partitionedInto/disjoint-decomposition`, `##thePartition/partition`, `##genlMt`, `##argXIsa/nth-domain` (where X is a digit), `##argXgenls/nth-domain-subclass-of` (where X is a digit), `##arity/function-arity/relation-arity`, `##resultIsa/range`, `##resultGenls/range-subclass-of`
- **Ground facts** are any statement without a variable.
- **Implications** include any non-ground statement that has an `##implies` (note that a ground statement that contains an `##implies` is counted as a ground statement)
- **Non-ground, non-implications** are statements that contain variables but not an implication.

This categorization is imperfect, but it is easy to implement and was applicable to both of the crisis management systems developed during the HPKB project.

The structural statements have an intuitive status in most systems: for SNARK the structural information is

sort information, for Cyc the structural information is called definitional, and for description logic systems the structural relations are usually called *concept constructors*. The statements with implications are rules. Ground facts often represent knowledge that can be found in an almanac or database.

A weakness of this categorization is that it counts many statements as ground statements even though they are not actually ground. For example, the statements involving `template-slot-value`, and `#$relationAllExists` are counted as ground. Further refinement to this categorization is left open for future work.

The axiom categorization scheme gave us an empirical tool to compare content across the two systems developed in the project. We would welcome proposals from the theoretical KR community, detailing more systematic ways to measure the competence of a large KB.

5. STANDARDS

Having a standard syntax is a necessity, but standard syntax plays a relatively small role in addressing the practical challenges facing the knowledge engineer. There is a need to move from an emphasis on standards of syntax, or on defining a precise semantics for tiny theories, to standard large theories and style guides for axiom writing.

For example, the subclass relationship can be either stated as

1. *(subclass-of A B)*, or as
2. *(=> (A ?x) (B ?x))*

Both of these forms are ANSI KIF. The first form uses *subclass-of* as a relation to compactly encode information that could also be written as in Form 2. The first form also has the advantage that a reasoner supporting taxonomic inference can take advantage of this form, which can be quite difficult for the second form.

As another example, consider three commonly used ways to specify the type information of variables in an axiom: (1) using ANSI KIF-style typed quantifiers, (2) using *instance-of* relations, or (3) using the class as a relation. Here is an example axiom encoded in these three forms:

```
(forall ((?x action)
         (?y action)
         (?z country))
    (=>
       (and
          (may-cause ?x ?y)
          (performed-by ?x ?z)
          (maleficiary ?y ?z))
          (risk-of-action
              ?x ?z ?y)))

(forall (?x ?y ?z)
    (=>
       (and
          (instance-of ?x action)
          (instance-of ?y action)
          (instance-of ?z country)
          (may-cause ?x ?y)
          (performed-by ?x ?z)
          (maleficiary ?y ?z))
       (risk-of-action ?x ?z ?y)))

(forall (?x ?y ?z)
    (=>
       (and
          (action ?x)
          (action ?y)
          (country ?z)
          (may-cause ?x ?y)
          (performed-by ?x ?z)
          (maleficiary ?y ?z))
       (risk-of-action ?x ?z ?y))
```

One additional factor might be that may-cause and risk-of-action could be defined as referring to types of actions rather than instances in different knowledge bases.

These three forms are equivalent and follow the ANSI KIF standard. In spite of the standard, people come up with sufficiently different ways to write axioms to make the knowledge exchange difficult. Therefore, the standards must be accompanied by a style guide before they can enable knowledge exchange. In the above example, the style guide could require that the type information for axioms should always be stated in the quantifier specification.

6. USING A VERY EXPRESSIVE REPRESENTATION

Expressive representations enable a degree of generality and reuse not possible with more restricted representations. Because of interactions among axioms, the inference time can become very high. The most general and reusable theory is not useful if inference on those theories is not tractable for your inference engine. Some ways of addressing this problem are by partitioning the KB into modules to isolate the interactions among axioms, and by compiling knowledge by hand into more efficient representations.

One team had the goal ot keeping the inference time for answering a question to less than 2 minutes. If all the axioms were loaded at the same search space, it was not possible to meet this requirement. Therefore, we *modularized* the KB to limit the interactions among axioms and achieve the desired response time. This problem would have been less critical had we limited the representation to horn clauses.

KB modularization means dividing the content of a KB into conceptual partitions that serve the basis for KB development and inference. We experimented with two ways to modularize a KB: subject based and task based. A *subject-based modularization* organizes a KB by subject area and can enable easier sharing and development of KB content. A subject area can be assigned to a knowledge engineer to direct its development. While reusing a KB, one can select a KB in the subject area of interest. A *task-based modularization* organizes a KB by the rules and individuals that are relevant to a task, thus significantly reducing the search space. The class, function, and relation definitions do not affect the search space, and therefore need not be modularized to speed up inference.

Modularization of a KB based on the subject-based criteria and the task-based criteria can be different and can coexist. We used both subject-based and task-based modularization during the project. For example, three major subject areas covered in our KB are *actions, agents,* and *interests.* We also created task-specific partitions in the KB based on specific parameterized questions (PQs). For example, for answering questions about interaction between interests and actions, there was no need for knowledge about specific terrorist groups in the KB that were kept in a separate partition. The approach to modularization described here was clearly engineering driven, and better principles to arrive at the modularization are needed. Techniques to develop modules for a KB in a way that isolates independent reasoning chains are clearly of special importance.

7. ISSUES IMPEDING PROGRESS

Inference engine performance is one crucial technical issue. While it is not easy to develop inference modules for very expressive features, it is incredibly hard to get those modules to perform well.

Despite the program's name, execution speed was not an issue under investigation in HPKB. Many researchers have studied algorithms, speed, and complexity. HPKB was extremely important because it focused on content. Much research on inference performance has not been undertaken in the context of practical reasoning on large knowledge bases. The challenge now is to focus on merging research on creating and reasoning with large knowledge bases with research on inference performance.

The most important nontechnical issue is research parochialism. The need to "own" a language, ontology, theory, or protocol is very powerful, whether in terms of building a research identity or a commercial base. However, this fragmentation is hampering progress.

Allen's seminal work (Allen, 1984) (Allen, 1994) on representing temporal knowledge is a good example of the kind of results that we need, and it is also well referenced and adopted in the applied AI community. Allen's work identified the primitives necessary to represent a sufficiently large class of temporal information and proposed inference procedures. If we could do the same for other domains such as actions, space, and causality, etc, it would greatly speed the practical KB construction. It is also the case that careful theoretical work has been done in these areas but may not be well known or adopted in the applied AI community. This work includes (Cohn et al., 1997), (Giunchiglia & Lifschitz, 1998), (Giunchiglia & Lifschitz, 1999), (Lifschitz, 1987), (McCain & Turner, 1997).

The KR community is still theoretically focused. Few people are interested in working on creating KB content. The time is right for a new focus on practical KB content creation.

Acknowledgments

We wish to acknowledge our DARPA sponsor, Murray Burke, for funding and guiding this work. We also wish to acknowledge the essential contribution of Robert Schrag at IET, who specified the Challenge Problem that made this research possible. Cleo Condoravdi provided a very helpful review of the paper.

References

Allen, J. (1984). "Towards a General Theory of Action and Time", Artificial Intelligence 23, pp 123-154.

Allen, James and George Ferguson (1994). "Actions and Events in Interval Temporal Logic", Journal of Logic and Computation 4, 531-579.

Booch, G. (1994). Object-Oriented Analysis and Design With Applications, Addison-Wesley

Chaudhri, V., J. Lowrance, J. Thomere, M. Stickel, and R. Waldinger (2000). Ontology Construction Toolkit. Artificial Intelligence Center, Technical Report.

Cohen, P, V. Chaudhri, A. Pease, and R. Schrag (1999). "Does Prior Knowledge Facilitate the Development

of Knowledge Based Systems", Proceedings of AAAI-99.

Cohen, P., R. Schrag, Jones, A. Pease, Lin, Starr, Gunning, and Burke (1998). "The DARPA High Performance Knowledge Bases Project", AI Magazine, Vol. 19 No.4, Winter.

Cohn, A., B. Bennet, J. Gooday, and N. Gotts (1997). Representation and Reasoning with Qualitative Spatial Relations about Regions. http://www.scs.leeds.ac.uk/spacenet/leedsqsr.html

Cycorp (1998). "Features of the CycL Language", on-line report at http://www.cyc.com/cycl.html .

Genesereth, M., and R. Fikes (Editors) (1992). Knowledge Interchange Format, Version 3.0 Reference Manual, Computer Science Department, Stanford University, Technical Report Logic-92-1, June.

Giunchiglia, E., and V. Lifschitz (1998). An action language based on causal explanation: preliminary report. In Proceedings AAAI-98, pp. 623-630.

Giunchiglia, E., and V. Lifschitz (1999). "Action Languages, Temporal Action Logics and the Situation Calculus". In Working Notes of the IJCAI-99 Workshop on Nonmonotonic Reasoning, Action, and Change.

HPKB Web (1999). "HPKB Web Site", http://projects.teknowledge.com/HPKB/

HPKB Pubs (1999). "HPKB Publications Page", http://projects.teknowledge.com/HPKB/Publications.html

Lenat, D., 1995, "Cyc: A Large-Scale Investment in Knowledge Infrastructure". Communications of the ACM 38, no. 11, November.

Lifschitz, V. (1987). "Formal Theories of Action". The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Workshop. Los Altos, CA: Morgan Kaufmann Publishers.

McCain and Turner (1997), "Causal Theories of Action and Change", Proceedings of AAAI-97, pp 460-465.

Schrag, R. (1999:1), email communication.

Schrag, R. (1999:2). "HPKB Year 2 Crisis Management, End-to-end Challenge Problem Specification", Version 1.2, February 5, Information Extraction and Transport, Inc. and Pacific-Sierra Research Corp. Rosslyn, VA. http://www.iet.com/Projects/HPKB/Y2/Y2-CM-CP.doc

Stickel, M., R. Waldinger, M. Lowry, T. Pressburger, and I. Underwood (1994). "Deductive Composition of Astronomical Software from Subroutine Libraries", in Proceedings of the Twelfth International Conference on Automated Deduction (CADE-12), June, 341-355.

Teaching Knowledge Representation: Challenges and Proposals

Leora Morgenstern
IBM T.J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532
leora@watson.ibm.com

Richmond H. Thomason
AI Laboratory
University of Michigan
Ann Arbor, MI 48109–2210
rich@thomason.org

Abstract

This position paper was prepared for a panel to be held at KR2000. The authors hope that it will help to stimulate discussion, planning, and cooperation concerning teaching issues in the KR community.

1 INTRODUCTION AND MOTIVATION

Knowledge Representation has played a crucial role in the development of Artificial Intelligence, and remains one of the strongest subfields of AI. From the earliest days of AI, leading researchers such as (McCarthy 1959), (McCarthy & Hayes 1969), and (Minsky 1969) have argued that in order for a program to act intelligently, it must have sophisticated methods of representing and reasoning with knowledge. The contributions of KR research—e.g., the use of formal logic for representing knowledge, automated theorem proving techniques, logic programming, semantic networks, and inheritance techniques—have been at the forefront of the AI intellectual scene. Whatever one may say about the value of human-level AI as a research goal,[1] KR continues to be a crucial source of ideas for AI technology. It has influenced the activity at AI research and development laboratories, and many of its ideas have spread to the general computer science community. Among the best known examples of this phenomenon are object-oriented programming languages (such as C++ and Java), to which inheritance techniques are central, and program specification languages which are modelled after Prolog.

Despite the centrality of KR to AI and Computer Science, it remains somewhat marginalized in the CS and AI curriculum. KR is rarely required in the CS curriculum; indeed, both graduate and undergraduate AI students can and often do complete their studies without taking a course in KR. Many universities don't even offer a course in KR, although departments which offer a two-semester course in AI often teach a good deal of KR in the second semester. There are few textbooks in KR, although there are KR-related texts like (Davis 1990) and (Sowa 1999) that concentrate on specific approaches and subtopics.

KR's lack of visibility in the curriculum is bad for CS and AI students, bad for CS departments, and bad for the future of KR itself.

A good KR course can be an intellectual eye-opener. It offers students an excellent—often, a unique—opportunity to see how ideas from computer science interact with system development. Typical CS students come to their first AI course with good programming skills and a fair amount of experience in small-scale projects. But without KR, they are unlikely to understand the importance of analyzing the reasoning task at hand, of separating out declarative elements, or of modularizing system design. KR not only serves as an important subject in its own right, introducing the research topics covered in KR, but often provides the transition from a naive to a more sophisticated understanding of system design.

KR applications abound with illustrations of this point. In developing an expert system, it is important to separate out the KR issues from software engineering issues. When designing educational software, it is crucial to correctly model the underlying domain knowledge so that one can construct coherent and useful explanations for the users. In developing a large-scale knowledge base, one must tackle head-on issues of ontology development to ensure proper organization for efficient updating, retrieval, and inference.

KR provides a way of integrating formal representa-

[1] See (Nilsson 1995).

tion languages and ideas from theoretical CS with challenging applications. This point has been stressed by (Sandewall 2000), who argues that the role of formal representation in AI and CS is akin to

> the role of calculus in engineering, that is, as a conceptual and notational tool that allows one to model phenomena in the world with precision, and to design software systems based on those models.

Applications and programming systems change at a relatively rapid rate, and are often best learned in the workplace. Basic scientific principles, on the other hand, have a longer horizon, and if they are not learned in an academic setting are unlikely to be acquired in the workplace. Computer *science* as a field presupposes that there are such basic principles, and that they are crucially important for computing technology. As one of the best ways of illustrating the importance of scientific ideas in software development, KR is important for a balanced and successful CS curriculum.

Anyone attending a KR meeting or reading a KR Proceedings doesn't need to be convinced of these things. But it may be important for the KR community to hone these arguments and practice them on colleagues, in an effort to secure a more favorable environment for teaching KR in universities. We need better publicity for the importance of KR in the curriculum, as well as better solutions to the challenges of teaching KR.

In the next sections, we first examine some of the major challenges for KR teachers, and propose some solutions; then we discuss issues of content and resources.

2 CHALLENGES OF TEACHING KR

Teaching KR can be an incredibly uplifting experience. There is great satisfaction in introducing students to a fundamentally different way of thinking about what they do—and KR, with its emphasis on formal methods of representation, does just that. At the same time, anyone who has taught KR on a regular basis probably has experienced some problems in the process. There are many potential difficuties facing KR teachers. Below, we single out four major challenges. We then discuss the relation between teaching KR and training users in the commercial world to use KR-based applications. The connection between these two activities suggests that developing improved methods for teaching KR can also facilitate the dissemination of KR within the commercial arena.

2.1 LACK OF STUDENT PREPARATION

Because the central idea of KR is the representation of knowledge in formal languages, students need a good grasp of logic to succeed in the study of KR. Specifically, students need to know first-order logic, the most important general-purpose representation and reasoning system. They also need to know basic metatheoretic techniques, including model theory and proof theory, preferably through a proof of the soundness and completeness of first-order logic. Moreover, they should be competent in translating at least simple natural language sentences to first-order logic and should have some understanding of the difficulties that translation from natural language to logic can present. Ideally, they should have some familiarity with other logics, such as modal, temporal, and dynamic logics; certainly first-order logic is the barest requirement.

Unfortunately, in our experience, many students come in with a smattering of logic, picked up, sometimes reluctantly, in a required introductory logic course somewhere along the way. Logic is seldom taught as a separate unit in the Computer Science curriculum, and content and standards of logic instruction differ widely outside of computer science. Many introductory courses concentrate mostly on propositional logic, and give students only a taste of first-order logic. Many courses are weak on metatheory. Students rarely learn how to translate between natural language and a formal language like first-order logic.

Meeting the Challenge:

There is no simple cure. It is clear that in the long term we ought to agitate for more training in logic, and for this training to occur early on in a student's career. High school, or even junior high school, is not too early to introduce students to both propositional and predicate logic. Logic should be emphasized in college as an important part of the core curriculum, and certainly should be required early on for any student majoring in computer science.

We don't imagine that waging this sort of campaign will be easy. But we should use the persuasive arguments that have been made by, among others, Robert Kowalski (Kowalski 1990, 1993), who has spoken and written at length about the close connection between writing clearly and being able to translate natural language into first-order logic. Kowalski argues that writing and rewriting one's thoughts in successively clearer drafts of English (or one's native natural language) brings one closer to the process of formalizing these ideas in logic. Making explicit and publicizing the close connection between training in logic and the

development of writing skills might help elevate logic's importance in the curriculum—not only in the CS curriculum but in the liberal arts and sciences curriculum. Our motto ought to be that training in logic develops strong and flexible minds; it should become a core part of the twenty-first century curriculum.

This is an ambitious goal, which will take time to implement. In the short term, we need quick solutions to the problems that arise when students are unprepared. Most KR teachers have probably experienced the fallout (both in terms of student attrition and students' complaints) from assigning the first homework requiring substantial formalization in logic. It is usually at this point that students realize how poorly prepared they are. How can we help them? We have four suggestions.

- *Mini-courses.* It is best if students realize quickly that they have to make up deficiencies; possibly this can be accomplished by making assignments available before class begins or on the first day of class. To bring students up to speed, intensive mini-courses, held either before a course starts or concurrently with the course at the beginning of the semester, can be very helpful. This is no panacea; many students need the time for difficult concepts and methodologies to sink in, and many don't realize how much they need this extra training. On the other hand, in our experience, logically disadvantaged students can profit from intensive immersion and practice.

- *Instructional software.* Some universities (Carnegie Mellon University, for one) have offered elementary logic instruction through tutoring software. Good textbooks with tutoring packages have been published; see especially (Barwise & Etchemendy 1999). Although software packages can provide useful instruction in the "art of logic," there is as far as we know no instructional software that deals with the much more challenging area of metatheory.

- *Tutoring and labs.* Some universities (for example, the City College of New York) have math labs where students can get free tutoring in remedial math, calculus, and courses like linear algebra and differential equations. The only way students can succeed is through extended practice; environments where drill is customary and pleasant are critical for successful logic instruction, especially for students who have been underexposed to logic. Once the importance of logic in the curriculum is established, it could become possible to argue successfully for logic labs.

  It might be difficult to argue successfully for such a use of university resources on the basis of a need in only one department, or even one school. But developments of this kind might be possible with more systematic cooperation between the various units across the university that are involved in logic instruction: Computer Science, Information Science, Mathematics, Linguistics, and Philosophy. This sort of cooperation is essential if we believe in creating a university climate in which KR instruction can flourish.

- *Written guidelines, examples, and texts.* There are many good logic textbooks, but it is hard to find one that does an adequate job of teaching formalization techniques. (Barwise & Etchemendy 1990) is one of the best recent introductions to formalizing natural language examples, with a range of examples and exercises, and extended discussion of the relation between natural language and logic. The Tarski's World domain used in the tutorial software that accompanies this textbook raises the issue of "micro-formalization"—of how to formalize microworlds in the Tarski's World domain. But the book offers little systematic help in macro-formalization, the art of formalizing extended, non-trivial domains. The most extended textbook discussions of these matters that we know of are (Genesereth & Nilsson 1987, Chapter 2) and the chapters on formalizing specific domains (e.g., Chapters 6 and 7) in (Davis 1990). (Davis 1999) offers a very useful outline of techniques for formalizing commonsense knowledge; a fleshed out outline would be an immense help. Sharing (on the web) some step-by-step approaches to macro-formalization would be very useful; ultimately, we need one or more published handbooks devoted to instruction in formalization techniques.

2.2 LACK OF MOTIVATION

There are two kinds of computer science students: those who want to program and build systems, and those who are interested in the more theoretical aspects of computer science. Students who want to build systems relate to what is immediately practical: programming courses, software engineering, robotics. They are excited by today's hot topics: the internet (and all the dot-com millionaires), e-commerce, computer animation. In some areas, there has been an application-driven shift away from KR methodology: natural language processing courses are more likely to

stress corpus-based and statistical techniques than the KR-based NLP techniques that were popular ten years ago.[2]

Of course, KR is not in the least irrelevant, though it may be seen as irrelevant for application-oriented students who crave immediate gratification. In fact, it is central to many of the most exciting new fields, such as knowledge management and information retrieval. We believe that the fundamental challenge faced by most industrial applications of AI is the problem of how to represent and reason with knowledge. Examples of such industry projects with which one author of this paper (Leora Morgenstern) is familiar include the development of an expert system for benefits inquiry in the medical insurance industry, the automatic configuration of life insurance and banking products; the development of an autonomous agent for searching out and understanding web pages, and diagnostic and helpdesk systems. There is only so far that clever programming techniques will get you. In the final analysis, if you get the knowledge representation part wrong in addressing these problems, the system won't work.

Meeting the Challenge:

This problem is largely a matter of presentation and public relations. Many practitioners of KR care as passionately about the importance of applications as they do about theory, and as KR matures, case studies of its usefulness are becoming more available. In teaching KR, we ought to begin with some of these case studies. Examples can be found in (Swift et al. 1994), (Moore 1995, Chapter 2), (Fikes & Farquhar 1997), (Morgenstern 1998), (Cui et al. 1999), (Brachman et al. 1999), and (Kautz & Selman 1999).

We also ought to put some serious thought into how KR can contribute to popular internet applications like information retrieval, text summarization, and e-commerce. One way of demonstrating KR's relevance would be to examine the inner workings of today's search engines and e-commerce applications, analyze these systems' shortcomings, and find ways in which adding KR methodologies could improve scale and performance. This exercise would be useful not only for teaching KR, but for the health and continued growth of KR as well. This point—the importance of applications for the good of the KR community— was argued in (Morgenstern 1998) for the particular field of nonmonotonic logic and inheritance; the argument needs to be made for KR as a whole.

[2]Compare (Jurafsky & Martin 2000), a textbook that is reasonably comprehensive, with, say, (Gazdar & Mellish 1989).

Theoretically minded computer science students are often attracted by the elegance and self-containment in theory of computation. This elegance is missing in typical introductory KR courses: there is little elegance, for instance, in introductions to first-order theories of planning and the use of frame axioms, where students laboriously grind through axioms to prove that Block A is still on Block B after Block C is moved to Block D. The better-developed areas of KR-related theory have as much elegance as any area of theory.[3] Even if theory can't be the focus of an introduction to KR, theory-minded students need to be made aware of the opportunities and achievements in this area.

2.3 SCARCITY OF TOOLS

Teaching KR could greatly benefit from a set of easy-to-use tools: e.g., tools for standard inheritance (classification and subsumption), tools for inheritance with exceptions, and logic programming tools. Such tools are useful for at least two reasons: First, using such tools allows students to easily see how KR can be used to solve interesting problems. Second, assigning meaningful and useful KR programming projects is almost impossible without tools. If students are required to develop the tools to implement basic KR methodologies from scratch, there will be little time to do the more exciting work that they need to do in order to remain interested in KR as a field.

Meeting the Challenge:

This challenge has already been partly met. Many teachers and research groups have generously devoted their time to making easy-to-use tools available to use: e.g., CMU's repository of KR software at

http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/kr/systems/0.html,

Vladimir Lifschitz's collection of planning software tools at

http://www.cs.utexas.edu/users/vl/teaching/planning.html,

which include, among others, a link to software for Datalog at

http://www.dbai.tuwien.ac.at/proj/dlv,

one to Smodels software at

http://www.tcus.hut.fi/Software/smodels,

and one to materials made available by Eric Sandewall's and Patrick Doherty's group at Linköping at

http://www.ida.liu.se/%7Epatdo/kplabwebsite/software.htm.

[3](Lifschitz 1991), (Kautz & Selman 1989), (Nebel 1990), (Kraus et al. 1990), (Hodges 1994), (Borgida & Patel-Schneider 1994), and (Fagin et al. 1995), are examples. There are, of course, many others.

Many readers will see important omissions in this list. Its partiality indicates another need: for readily available, up-to-date indices of tools that can be useful in teaching KR. Besides obtaining a reasonably complete list of available resources, there is a need for further work in this area. For example, despite the fact that polynomial time algorithms for inheritance with exceptions have been known for over a decade (Horty et al. 1990) and (Stein 1992), there are, as far as we know, no currently available web tools that students can use.

We can fix these problems with some time, effort, and money. Researchers' time and money are always in short supply, but the realization that the investment in time now will pay for itself in the future should help convince many of us to set aside time for this purpose. Money is also always in short supply. We could consider applying for grants from the National Science Foundation or other funding sources to enable the development and dissemination of these tools.

2.4 LACK OF A STANDARDIZED CURRICULUM

There is currently no standard curriculum for Knowledge Representation. This may be partly because there is no standard textbook. (And there may be no standard textbook because the field is so new and fast-moving.) The problem is compounded by the fact that there is so much material to teach, and so little time to do it in. (Unlike courses such as databases or theoretical computer science which often are given over two semesters, with plenty of time for the exploration of various topics, KR, if it is at all offered at a university, is a one-semester course.)

KR teachers have very different ideas of what a KR course should include. Should one concentrate on formalizations in first-order logic, focus on particular domains such as physical and spatial reasoning, integrate the material at the outset with practical applications, include semantic networks or Bayesian networks, include non-standard logics such as nonmonotonic logics, modal logics, and multi-valued logics?

Clearly it is impossible to cover all of these topics (with any degree of thoroughness) in one course, but some agreement would be helpful on what subset of these topics is essential for grounding students in KR.

Meeting the Challenge:

A standardized curriculum would help to develop tools and teaching materials, and a consensus on what belongs in the curriculum would help the KR community present a united front on the subjects they consider most representative and important. We address these issues in Section 3.

2.5 TEACHING KR VS. USING KR IN THE FIELD

It is interesting to note that many of the problems that arise in teaching KR at a university often occur in a strengthened form when practitioners use KR to solve problems in industry. Such an effort almost always involves a partnership between the researhers and developers of the KR solution, on the one hand, and the users of the solution, on the other hand.

For example, in developing an expert system for benefits inquiry in a medical insurance corporation, one of the authors (Leora Morgenstern) used as the central structure a semantic network, consisting mostly of a taxonomy of medical services, augmented by business rules. The population of the semantic network was a joint effort between her and a small group of employees in the insurance corporation. These employees had to receive a crash course in semantic networks. They had to be taught how to identify the core concepts in the domain (in this case, medical procedures, conditions, and settings), how to enumerate the nodes in the network, and how to organize the nodes. Once the system was developed and in use, the policy modifiers — those insurance company employees who change insurance policy rules — had to learn how to change the network by adding, deleting, and modifying nodes and rules. Teaching these employees how to use semantic networks was quite similar to teaching semantic networks in a classroom (although users of commercial systems are already quite motivated).

We are not suggesting that developing KR for commercial applications entails teaching a full-fledged KR course in the field. However, we do wish to stress the connection between the two activities. We believe that many of the general methodologies that we develop to bring KR students up to speed can also be used when developing methods to train commercial users of KR-based systems. Conversely, if we are poorly prepared to teach KR to unsophisticated students, it will be that much harder to effectively use KR in the commercial world.

The realization that teaching KR to university students is closely related to training customers to use commercial KR systems should serve as added motivation to solve the problems that we face.

3 COURSE ORGANIZATION: SYLLABUS, MATERIALS, AND SOURCES.

3.1 THE SYLLABUS

As we mentioned, there is currently no consensus on what belongs in a KR course. We make some suggestions in this section, in the hope that this will at least serve to stimulate discussion on the topic within the KR community.

We suggest that a KR course include at the minimum a review and discussion of first-order logic; an analysis of why first-order logic is not in itself sufficient for the KR enterprise, an example of an extension of first-order logic (e.g., modal logics or logics augmented by quotation or a nonmonotonic logic), semantic networks, inheritance with and without exceptions, temporal logic and planning, and a discussion of how KR systems have been used in real-world applications. Other topics which may be included, time permitting, include a more detailed discussion of modal logic, the study of some particular domains of commonsense reasoning such as spatial or physical reasoning, logic programming, specific nonmonotonic formalisms, explanation, and Bayesian nets.

Below is a possible syllabus for a KR course. We assume a fourteen-week semester with classes meeting twice a week, giving twenty-eight classes. Some topics are listed below, with the assumption that they will be gone through in this order with anywhere from 2-4 sessions on each topic.

1. Basic principles of KR and motivation. Case study of systems using KR, with demos. Case study of systems not using KR (like today's search engines); discussion of how KR methodologies could be used to improve them. Use these initial discussions for ideas for projects later on.

2. Basic first-order logic. Introduction/review, practice and more practice. Very brief overview of why first-order logic is not enough to represent general knowledge, and why modal logic and nonmonotonic logics will be needed.

3. Logic Programming methodology.

4. Semantic networks and examples of applications (e.g., from natural language processing).

5. Inheritance networks with exceptions, brief mention to nonmonotonic logics.

6. Temporal logic. The situation calculus, perhaps the event calculus, examples of planning.

7. Basics of modal logic. Epistemic logics; examples of applications from distributed computing (Fagin et al. 1995) ; Deontic logics.

8. Basics of nonmonotonic logics. Circumscription, default logic, autoepistemic logic. Nonmonotonic semantics (negation as failure) for logic programming.

9. Comparison with other formal methodologies such as probabilistic reasoning. Bayesian networks.

10. Using above techniques for applications. Detailed examples.

3.2 MATERIALS, TOOLS, AND SOFTWARE

A successful KR course needs to be firmly grounded in both theory and practice. As such, it would be helpful to provide students with both written materials and software tools.

Written Materials

As we have already said, there is a dearth of KR textbooks. Certainly no existing textbook covers, for example, the syllabus suggested in the previous section. Still, the available texts are useful as a starting point. They should be augmented, of course, with readings. The fifteen-year-old collection (Brachman & Levesque 1985), while of course out of date, is nevertheless important because it contains so many classic papers. In addition, there are many excellent recent papers on specific topics, and general encyclopedia articles that can give students good overviews (e.g., (Davis 2001a), (Hayes 1999) , and the older (Barr & Davidson 1981). As for newer collections, (Brachman et al. 1992) is helpful, as well, of course, as the KR Conference Proceedings from 1989 onwards. Teachers can of course compile lists of important readings covering topics not addressed in existing texts; such lists may consist both of hard-copy papers (distributed in class or put on reserve in the library) or links to on-line papers. It would be very useful to have a central website where such lists could be maintained; this would be an important step towards achieving consensus on a curriculum.

In addition to textbooks and papers, the authors have found that students often find slide presentations very helpful. We are not entirely comfortable with encouraging a reliance on quick summaries and easy sound bites; nevertheless, it is hard to argue with anything

that can help students plow their way through difficult material. There is a wealth of material on the web, including slides that have been used for tutorials at conferences and slides that have been used in individual classes. Examples include a collection of tutorials for description logics

http://www.cs.man.uk/~franconi/dl/course ;

sets of slides accompanying the introductory textbook (Poole et al. 1998)

http://www.cs.ubc.ca/spider/poole/ci/lectures/lectures.html),

and the websites of the authors' courses. A central website with links to these materials would be very useful.

Software Tools

Ideally, the software tools used for a KR course should meet the following requirements:

- they should be quick to learn and easy to install and use;
- they should be platform independent, or at least, be available on a variety of platforms and not require installation of too much accessory software;
- they should serve as the basis of useful and interesting applications; that is, students using the software ought to be able to develop non-trivial KR applications with relative ease;
- they should be neutral with respect to ongoing ideological wars in the KR community.

We don't know of an existing substantial library of software tools meeting the above requirements; we hope that the creation of such a library will occur sometime in the future. Such a library could include general theorem-proving tools including logic programming software and nonmonotonic theorem provers, semantic network tools, tools for inheritance with exceptions, Bayesian reasoning tools, and tools for temporal reasoning and planning.

One of the authors (Rich Thomason) has found the CLASSIC system to be an valuable tool in teaching KR. The clean design and excellent documentation of this system make it especially appropriate for teaching students; at the same time, it provides a suitable framework for serious representation projects. Often, these projects can be integrated into larger systems on which students are working. An online tutorial for CLASSIC is available at http://www.eecs.umich.edu/ rthomaso/kr/classic-tutorial.html

Of course, the need for a central website with pointers to useful KR software is just as great as (if not greater than) the need for an index of written materials.

4 FUTURE PLANS

We believe that a centralized effort is necessary in order to strengthen the role of KR in the AI and CS curriculum. Specifically, we believe that it may be helpful for the KR organization to become more active in supporting and promoting the teaching of KR. The following possibilities have occurred to us; others may emerge from discussion at KR2000.

1. Establishing a website devoted to teaching KR. Such a website could include the libraries for written materials and software tools discussed in the previous section.
2. Gathering information concerning perceived problems and solutions. It might be useful to conduct some sort of survey of those teaching KR, to find out the problems that they have encountered and how they have tried to address these problems. The results of this survey could be shared with the KR community. We have doubtless faced many of the same problems, and sharing solutions could save time and energy.
3. Preparing a position statement on the role of KR in the Computer Science curriculum, and sharing it with the KR community. Such a position paper could be used—in whole or in part—by KR faculty who need to strengthen the role of KR in their departments, and could also be useful in writing grant applications requesting funding for the development of KR software tools.

All of this will involve a good deal of work. Of course, we can't realistically expect more to be done than individuals can manage to do on an unpaid basis. Still, historically, much academic progress has been made possible by volunteer labor. The usual machinery that has facilitated and leveraged this labor—the formation of a small committee or task force in charge of these issues, a short-term mandate to construct a website of the sort discussed above (along with a way to count it as a publication)—should stand us in good stead here. We expect that the work involved will strengthen the position of KR in the AI and CS communities.

References

[1] Avron Barr and James E. Davidson. Representation of knowledge, vol. 1. In Avron Barr and

Edward A. Feigenbaum, editors, *The Handbook of AI*, pages 141–222. HeurisTech Press, Stanford, California, 1981. Co-authors: Robert Filman, Douglas Appelt, Anne Gardiner, and James Bennett.

[2] Jon Barwise and John Etchemendy. *Language, Proof, and Logic.* CSLI Publications, Stanford, California, 1999.

[3] Alex Borgida and Peter Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[4] Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter, editors. *Knowledge Representation.* The MIT Press, Cambridge, Massachusetts, 1992.

[5] Ronald J. Brachman, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lori A. Resnik. "reducing" CLASSIC to practice: Knowledge representation theory meets reality. *Artificial Intelligence*, 114(1–2):203–237, 1999.

[6] Ronald Branchman and Hector Levesque, editors. *Readings in Knowledge Representation.* Morgan Kaufmann, San Francisco, 1985.

[7] Baoqui Cui, Terrance Swift, and David S. Warren. A case study in using preference logic grammars for knowledge representation. In Michael Gelfond, Nicola Leone, and Gerald Pfeifer, editors, *Lecture Notes in Artificial Intelligence 1730: Logic Programming and Nonmonotonic Reasoning*, pages 206–220, Berlin, 1999. Springer-Verlag.

[8] Ernest Davis. *Representations of Commonsense Knowledge.* Morgan Kaufmann, San Francisco, 1990.

[9] Ernest Davis. Guide to axiomatizing domains in first-order logic. *Electronic Newsletter on Reasoning about Actions and Change*, 99002, 1999. http://www.etaij.org/rac/.

[10] Ernest Davis. Knowledge representation. In Neil J. Smelser and Paul B. Baltes, editors, *The International Encyclopedia of the Social and Behavioral Sciences.* Elsevier, Amsterdam, 2001.

[11] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge.* The MIT Press, Cambridge, Massachusetts, 1995.

[12] Richard Fikes and Adam Farquhar. Large-scale repositories of highly expressive reusable knowledge. Unpublished manuscript, Knowledge Systems Laboratory, Stanford University, 1997.

[13] Gerald Gazdar and Chris Mellish. *Natural Language Processing in Prolog: An Introduction to Computational Linguistics.* Addison-Wesley Publishing Co., Reading, Massachusetts, 1989.

[14] Michael Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann, San Mateo, California, 1987.

[15] Patrick Hayes. Knowledge representation. In Robert A. Wilson and Frank C. Keil, editors, *MIT Encyclopedia of the Cognitive Sciences.* The MIT Press, Cambridge, Maassachusetts, 1999.

[16] Wilfrid Hodges. Logical features of Horn clauses. In Dov Gabbay, Christopher Hogger, and J.A. Robinson, editors, *The Handbook of Logic in Artificial Intelligence and Logic Programming, Volume I*, pages 449–503. Oxford University Press, Oxford, 1994.

[17] John F. Horty, Richmond Thomason, and David Touretzky. A skeptical theory of inheritance in nonmonotonic semantic networks. *Artificial Intelligence*, 42:311–349, 1990.

[18] Daniel Jurafsky and J. Martin. *Speech and Language Processing.* Prentice Hall, Englewood Cliffs, New Jersey, 2000.

[19] Henry Kautz and Bart Selman. Hard problems for simple default logics. In Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter, editors, *KR'89: Principles of Knowledge Representation and Reasoning*, pages 189–197. Morgan Kaufmann, San Mateo, California, 1989.

[20] Henry Kautz and Bart Selman. Unifying SAT-based and graph-based planning. In Thomas Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 318–325, San Francisco, 1999. Morgan Kaufmann.

[21] Robert A. Kowalski. English as a logic programming language. *New Generation Computing*, 8(2):91–93, 1990.

[22] Robert A. Kowalski. An undergrauate degree in practical reasoning. *Journal of Logic and Computation*, 3(3):227–229, 1993.

[23] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 14(1):167–207, 1990.

[24] Vladimir Lifschitz. Towards a metatheory of action. In James Allen, Richard Fikes, and Erik Sandewall, editors, *KR'91: Principles of Knowledge Representation and Reasoning*, pages 376–386. Morgan Kaufmann, San Mateo, California, 1991.

[25] John McCarthy. Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pages 75–91, London, 1959. Her Majesty's Stationary Office.

[26] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

[27] Marvin Minsky. *Semantic Information Processing*. The MIT Press, Cambridge, Massachusetts, 1969.

[28] Johanna Moore. *Participating in Explanatory Dialogues*. The MIT Press, 1995.

[29] Leora Morgenstern. Inheritance comes of age: Applying nonmonotonic techniques to problems in industry. *Artificial Intelligence*, 103(1–2):237–271, 1998.

[30] Bernhard Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43(2):235–249, 1990.

[31] Nils J. Nilsson. Eye on the prize. Available at www.robotics.stanford.edu/ nilsson., 1995.

[32] David Poole, Alan Mackworth, and Randy Goebel. *Computational Intelligence: A Logical Approach*. Oxford University Press, New York, 1998.

[33] Erik Sandewall. On the methodology of research in knowledge representation and common-sense reasoning. In Jack Minker, editor, *Logic-Based Artificial Intelligence*. Kluwer Academic Publishers, Dordrecht, 2000. Forthcoming.

[34] John F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Thomson Learning, Stamford, Connecticut, 1999.

[35] Lynn Andrea Stein. Resolving ambiguity in nonmonotonic inheritance hierarchies. *Artificial Intelligence*, 55(2–3):259–310, 1992.

[36] T. Swift, C. Henderson, R. Holberger, J. Murphey, and E. Neham. Cctis: An expert transaction processing system. In *Proceedings of the Sixth Conference on Industrial Applications of Artificial Intelligence*, pages 131–140, 1994.

Author Index

Amir, Eyal, 389

Baader, Franz, 261, 297
Balbiani, Philippe, 378
Baral, Chitta, 311
Ben Amor, N., 235
Benferhat, S., 235
Bennett, B., 15
Besnard, Philippe, 401
Biso, Alessandro, 435
Bloch, Isabelle, 247

Calvanese, Diego, 176
Chalupsky, Hans, 471
Chaudhri, Vinay, 717
Cheng, Shifu, 647
Chomicki, Jan, 121
Ciocoiu, Mihai, 539
Clark, Peter, 591
Cohn, A. G., 15
Condotta, Jean-François, 571
Cristani, M., 15
Cumby, Chad, 425

De Giacomo, Giuseppe, 176
Denecker, Marc, 74
Dimopoulos, Yannis, 53
Dubois, D., 235

Eiter, Thomas, 62
El Fattah, Yousri, 213

Fargier, Hélène, 445
Farquhar, Adam, 717
Fikes, Richard, 483

Galton, Antony, 26
Geffner, Héctor, 235, 667
Ginsberg, Matthew L., 690
Giunchiglia, Enrico, 657

Haarslev, Volker, 273
Hahn, Udo, 601
Horrocks, Ian, 285

Iocchi, Luca, 678
Ismail, Haythem O., 355

Jamil, Hasan M., 611
Janhunen, Tomi, 411
Jeansoulin, Robert, 505

Küsters, Ralf, 261, 297
Konieczny, Sébastien, 135

Lafage, Céline, 457
Lamperti, Gianfranco, 333
Lang, Jérôme, 445, 457, 625
Lehmann, Daniel, 153
Lehmann, Fritz, 717
Lenzerini, Maurizio, 176
Lespérance, Yves, 527
Levesque, Hector J., 527
Li, Bin, 647
Liberatore, Paolo, 145
Lifschitz, Vladimir, 85
Lin, Fangzhen, 167
Lobo, Jorge, 121
Lukasiewicz, Thomas, 62

Marek, Victor W., 74
Marquis, Pierre, 445, 625
Martín, Mario, 667
Massacci, Fabio, 186
Maynard-Reid II, Pedrito, 153
McCarthy, John, 519
McGuinness, Deborah L., 483
McIlraith, Sheila, 311, 389
Molitor, Ralf, 297
Möller, Ralf, 273
Montanari, Angelo, 547
Morgenstern, Leora, 725
Morris, Paul, 580
Morris, Robert A., 580
Mota, Edjard, 366

Naqvi, Shamim, 121
Nardi, Daniele, 678
Nau, Dana S., 539
Nebel, Bernhard, 53
Niemelä, Ilkka, 411

Osmani, Aomar, 378

Pagnucco, Maurice, 527
Papini, Odile, 505
Parkes, Andrew J., 690
Pease, Adam, 717
Peot, Mark Alan, 213
Pino-Pérez, Ramón, 637
Policriti, Alberto, 547
Porter, Bruce, 591
Prade, H., 235

Reiter, Ray, 99
Reynolds, Robert G., 494
Rice, James, 483
Rosati, Riccardo, 678
Rossi, Francesca, 435
Roth, Dan, 425
Rychtyckyj, Nestor, 494

Schaerf, Marco, 145
Schaub, Torsten, 401
Schulz, Stefan, 601
Shapiro, Steven, 527
Shapiro, Stuart C., 355
Simons, Patrik, 411
Slanina, Matteo, 547
Son, Tran Cao, 311
Sperduti, Alessandro, 435
Stell, John G., 38

ten Teije, Annette, 323
Thielscher, Michael, 109
Thomason, Richmond H., 702, 725
Thompson, John, 591
Tobies, Stephan, 285
Toni, Francesca, 53
Truszczyński, Miroslaw, 74

Uzcátegui, Carlos, 637

van Harmelen, Frank, 323
Vardi, Moshe Y., 176
Verberne, Alan, 323
Vidal, Thierry, 559
Voronkov, Andrei, 198

Wassermann, Renata, 345
Wilder, Steve, 483
Wolter, Frank, 3
Wujia, Zhu, 647
Würbel, Eric, 505

Yelland, Phillip M., 225
You, Jia-Huai, 411

Zakharyaschev, Michael, 3
Zanella, Marina, 333
Zhaohui, Zhu, 647

PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING: PROCEEDINGS OF THE SEVENTH INTERNATIONAL CONFERENCE

Edited by Anthony G. Cohn (University of Leeds, UK), Fausto Giunchiglia (University of Trento and ITC-irst, Italy) and Bart Selman (Cornell University, USA)

The Knowledge Representation (KR) conferences have established themselves as the leading forum for timely, in-depth presentation of progress in the theory and principles underlying the representation and computational manipulation of knowledge.

The papers in this volume, which are twice as long as papers in general AI conferences, have passed a stringent review process. The topics covered include spatial, temporal, automated, and nonmonotonic reasoning; description logics, representation of action, and representation formalisms; integration of knowledge sources, learning and decision making, and knowledge engineering; applications and systems, diagnosis, planning, and uncertainty. Within these topics, treatments range from the abstract specification of algorithms and their computational complexity to the analysis of implemented systems.

These proceedings are an essential reference volume for researchers and students interested in a detailed view of this key research area.

Additional Titles of Interest from Morgan Kaufmann

FOUNDATIONS OF GENETIC ALGORITHMS, Volumes 1–5 edited by Gregory J. E. Rawlins, Darrell Whitley, Michael D. Vose, Richard J. Belew, Wolfgang Banzhaf, and Colin Reeves

ARTIFICIAL INTELLIGENCE: A NEW SYNTHESIS by Nils J. Nilsson

GENETIC PROGRAMMING: AN INTRODUCTION by Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone

READINGS IN AGENTS edited by Michael N. Huhns and Muninder P. Singh

UNCERTAINTY IN ARTIFICIAL INTELLIGENCE: PROCEEDINGS OF THE SEVENTH – FIFTEENTH CONFERENCES (1991–1999)

INTRODUCTION TO KNOWLEDGE SYSTEMS Mark Stefik

ELEMENTS OF MACHINE LEARNING Pat Langley

PLANNING AND CONTROL Thomas L. Dean and Michael P. Wellman

REPRESENTATIONS OF COMMON SENSE KNOWLEDGE Ernest Davis

PRINCIPLES OF SEMANTIC NETWORKS: EXPLORATIONS IN THE REPRESENTATION OF KNOWLEDGE edited by John Sowa

READINGS IN PLANNING edited by James Allen, James Hendler, and Austin Tate

ISBN 1-55860-690-4
ISSN 1046-9567
Artificial Intelligence

Morgan Kaufmann Publishers
340 Pine Street, 6th Floor
San Francisco, CA 94104
http://www.mkp.com

ISBN 1-55860-690-4