KR2002

Principles of Knowledge Representation and Reasoning

Proceedings of the Eighth International Conference

Toulouse, France
April 22–25, 2002

Edited by:

Dieter Fensel

Fausto Giunchiglia

Deborah McGuinness

Mary-Anne Williams

Principles of Knowledge Representation and Reasoning:

Proceedings of the
Eighth International
Conference
(KR2002)

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

Principles of Knowledge Representation & Reasoning: Proceedings of the Seventh International Conference (KR 2000)
Edited by Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman ISBN 1-55860-690-4

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 6–10, December 1997–December 1999
Edited by Michael P. Wellman

Principles of Knowledge Representation and Reasoning:

Proceedings of the
Eighth International
Conference
(KR2002)

Edited by

Dieter Fensel
(Vrije Universiteit, The Netherlands)

Fausto Giunchiglia
(University of Trento, Italy)

Deborah McGuinness
(Stanford University, USA)

Mary-Anne Williams
(The University of Newcastle, Australia)

Toulouse, France
April 22–25, 2002

Sponsored by KR, Inc.

Morgan Kaufmann Publishers
San Francisco, California

These proceedings were managed and produced for the organizers
of the KR2002 conference by Professional Book Center, Denver, Colorado.

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

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

ISBN 1-55860-847-8
ISSN 1046-9567

02 03 04 05 4 3 2 1

Contents

NONMONOTONIC REASONING

DESCRIPTION LOGICS

BELIEF REVISION

COMPLEXITY ISSUES II

REASONING ABOUT ACTION I

MERGING AND INTEGRATION II

ONTOLOGICAL ISSUES

UNCERTAINTY AND VAGUENESS

REASONING ABOUT ACTION II

APPLICATIONS AND IMPLEMENTATION

TEMPORAL REASONING

COMPLEXITY ISSUES III

KNOWLEDGE REPRESENTATION AND PLANNING (JOINT SESSION WITH AIPS)

INVITED SPEAKERS AND PANEL ABSTRACT

Preface

This volume contains the papers presented at the Eighth Conference on Principles of Knowledge Representation and Reasoning. Knowledge Representation and Reasoning (KR&R) is a vibrant and exciting field of human endeavor. KR&R techniques are key drivers of innovation in computer science, and they have lead to significant advances in practical applications in areas from Artificial Intelligence to Software Engineering.

The Knowledge Representation and Reasoning 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. Papers presented at KR2002 emphasize the importance of knowledge representation and reasoning in a wide range of applications including intelligent agents and the Semantic Web.

The traditional high standard of the conference has been maintained. We received 161 extended abstracts from 34 countries on 4 continents. We were able to accept 56 excellent papers for presentation.

The conference will also be significantly enhanced and enriched by the presentations of our Invited Speakers who are all pioneers in the field: Peter Gärdenfors presents a talk entitled "The Role of Higher Order Similarity in Induction and Concept Formation," Jim Hendler "The Semantic Web: KR's Worst Nightmare?" and Bernhard Nebel "The Philosophical Soccer Player."

We co-located with the following related workshops and conferences: the Sixth International Conference on Artificial Intelligence Planning and Scheduling Systems (AIPS2002), the 2002 International Workshop on Description Logics (DL2002), the Conference on Electronic Commerce (CollECTeR Europe 2002), the Workshop on Formal Ontology, Knowledge Representation and Intelligent Systems for the World Wide Web (SEMWEB), the Ninth International Workshop on Knowledge Representation Meets Databases (KRDB-2002), Workshop on Logics for Agent-Based Systems (LABS), and the Ninth International Workshop on Non-Monotonic Reasoning (NMR'2002).

Following the successful overlap of KR2000 and AIPS2000 in Breckenridge, Colorado, USA, we organized a joint session with AIPS2002. This will lead to further cross-fertilization and collaboration between the planning and knowledge representation research communities.

Dieter Fensel (Program Chair)

Fausto Giunchiglia (Conference Chair)

Deborah McGuinness (Program Chair)

Mary-Anne Williams (Program Chair)

Acknowledgments

KR2002 would not have been possible without the generosity of a large number of people. We would like to thank our outstanding program committee who performed the task of reviewing the 161 extended abstracts.

Program Committee Members

Franz Baader, RWTH Aachen

Fahiem Bacchus, University of Toronto

Lawrence W. Barsalou, Emory University

Salem Benferhat, Université Paul Sabatier

Brandon Bennett, University of Leeds

Gerhard Brewka, University of Leipzig

Marco Cadoli, Università di Roma

Marie-Odile Cordier, Université Rennes

Adnan Darwiche, University of California

Ernest Davis, New York University

Rina Dechter, University of California

Stefan Decker, Stanford University

Luis Farinas del Cerro, Université Paul Sabatier

Patrick Doherty, Linköping University

Didier Dubois, Université Paul Sabatier

Thomas Eiter, Vienna University of Technology

Hélène Fargier, Université Paul Sabatier

Richard Fikes, Stanford University

Tim Finin, University of Maryland

Antony Galton, University of Exeter

Hector Geffner, Universitat Pompeu Fabra

Enrico Giunchiglia, Università di Genova

Carole Goble, University of Manchester

Asuncin Gómez-Pérez, Univ. Poli. de Madrid

Georg Gottlob, University of Vienna

Nicola Guarino, LADSEB-CNR

Joseph Halpern, Cornell University

Frank van Harmelen, Vrije Univ. Amsterdam

Pat Hayes, University of West Florida

Ian Horrocks, University of Manchester

Anthony Hunter, University College London

Henry Kautz, University of Washington

Mark Keane, University College Dublin

Jana Koehler, IBM, Zurich

Hector Levesque, University of Toronto

Paolo Liberatore, Università di Roma

Vladimir Lifschitz, University of Texas at Austin

Fangzhen Lin, Hong Kong Univ. Sci. & Tech

David Makinson, Kings College

David McAllester, AT&T

John McCarthy, Stanford University

Sheila McIlraith, Stanford University

Robert A. Meersman, Vrije Universiteit Brussel

Leora Morgenstern, IBM Hawthorne

Ryszard Michalski, George Mason University

John-Jules Meyer, Utrecht University

Daniele Nardi, Università di Roma

Ilkka Niemela, Helsinki University of Technology

Lin Padgham, RMIT

Simon Parsons, University of Liverpool

Pavlos Peppas, Patras University

Ramon Pino Perez, Universidad de Los Andes

David Poole, University of British Columbia

Gregory Provan, Rockwell Scientific

Henri Prade, Université Paul Sabatier

David Randell, Imperial College

Stuart Russell, University of California, Berkeley

Marco Schaerf, Università di Roma

A. Th. Schreiber, University of Amsterdam

Bart Selman, Cornell University

Stuart C. Shapiro, SUNY Buffalo

Yoav Shoham, Stanford University

Liz Sonenberg, University of Melbourne

John F. Sowa

Rudi Studer, University of Karlsruhe

Michael Thielscher, Dresden Univ. of Technology

Rich Thomason, University of Michigan

Pietro Torasso, Università di Torino

Mirek Truszczynski, University of Kentucky

Toby Walsh, The University of York

Christopher A. Welty, Vassar College

Brian Williams, MIT

Frank Wolter, University of Leipzig

Mike Wooldridge, University of Liverpool

The following guest reviewers assisted the program committee.

Grigoris Antoniou
Alessandro Artale
Philippe Balbiani
Sean Bechhofer
Philippe Besnard
Marcus Bjäreland
Blai Bonet
Alex Borgida
Stefano Borgo
Jeen Broekstra
Diego Calvanese
Claudio Castellini
Oscar Corcho
Stephan Chalup
Samir Chopra
Luca Compagna
Francesco Donini
Yousri El Fattah
Giuseppe De Giacomo
Suzanne Embury
Jerome Euzenat
Mariano Fernandez-Lopez
Olivier Gasquet
Lluis Godo
Carla Gomes
Henrik Grosskreutz
Volker Haarslev
Patrik Haslum
Andrew Heathcote
Keijo Heljanko
Mark Hopkins
Luca Iocchi
Tomi Janhunen
Tommi Junttila
David Kinny
Michel Klein
Ralf Küsters
Jonas Kvarnström
Martin Lacher
Leonardo Lesmo
Faye Liu
Wei Liu
Peter Lucas
Carsten Lutz
Alexander Maedche
Mar Marcos
Maarten Marx
Claudio Masolo
Alessandro Masolo
Sergey Melnik
Rob Miller
Prasenjit Mitra
Yves Moinard
Massimo Narizzano
Abhaya Nayak
Alessandro Oltramari
Maurice Pagnucco
Jeff Z. Pan
Chris Partridge
David Poutakidis
Ren Quiniou
Jussi Rintanen
Riccardo Rosati
Anthony Roy
Marta Sabou
Ulrike Sattler
Eddie Schwalb
Krister Segerberg
John Slaney
Steffen Staab
Leon Sterling
Robert Stevens
Heiner Stuckenschmidt
Gerd Stumme
Andrzej Szalas
Armando Tacchella
Paolo Terenziani
Sergio Tessaris
John Thangarajah
Jin Tian
Stephan Tobies
Leon van der Torre
Carlos Uzcategui
Moshe Vardi
Bao Quoc Vo
Raphael Volz
Michael Winikoff
Wayne Wobcke
Stefan Woltran
Ronald R. Yager
Yuihui Yin
Michael Zakharyaschev
Dongmo Zhang

We would like to thank the rest of the KR2002 Organizing Committee: Bruce Porter, Workshops Coordination Chair; Peter Patel-Schneider, Publicity Chair; Laure Vieu, Treasurer; Andreas Herzig and Jérôme Lang, Local Arrangement Chairs. We are also exceedingly grateful for the efforts of the Local Organizing Committee and the Local WebMasters: Leila Amgoud, Nathalie Aussenac, Philippe Balbiani, Jean-Pierre Baritaud, Laurence Cholvy, Christophe Garion, Olivier Gasquet, Sébastien Konieczny, Céline Lafage, Dominique Longin, Nicolas Maudet, Jérôme Mengin, and Philippe Muller.

Special thanks to our Invited Speakers: Peter Gärdenfors, James Hendler and Bernhard Nebel. Thanks also to Christopher Welty who organized and chaired the panel entitled "Are Upper-Level Ontologies worth the effort?" Many thanks to the other panelists who helped make it a success: Pat Hayes, James Hendler, Nicola Guarino, and Fritz Lehmann.

We would like to thank the Newcastle Business School and the Business and Technology Research Lab at The University of Newcastle for funding the electronic reviewing process, as well as Daniel Le Berre, Janice Carrol, Greg Marr, Craig Murch, and Merrilyn Stone for providing the necessary support.

We are indebted to Jennifer Ballentine and the staff of Professional Book Center for their advice and assistance regarding the production of the proceedings.

We would also like to acknowledge the following organizations for their support: the Australian Research Council, DARPA, IRIT, Ontoweb, Mairie de Toulouse, ONERA Toulouse, Région Midi-Pyrénées, and Université Paul Sabatier.

Spatial Reasoning

A Logical Account of Perception Incorporating Feedback and Expectation

Murray Shanahan
Imperial College
Department of Electrical
Engineering,
Exhibition Road,
London SW7 2BT,
England.
m.shanahan@ic.ac.uk

Abstract

This paper presents the theoretical foundations of a vision system in which the effects of high-level reasoning percolate all the way down to influence the low-level processing of raw sensor data. This is achieved through the mechanisms of feedback and expectation. The main contribution of the paper is to present a formal framework, based on the abductive interpretation of sensor data, that incorporates the ideas of feedback and expectation in a way that marries them to logical reasoning. To enable this, two alternative measures of explanatory value are defined.

1 INTRODUCTION

Work in cognitive robotics has typically adopted one of two views of perception. According to the first view, perception is considered as a black-box process from the perspective of high-level cognition, whereby raw data is turned into fluents on demand, following the execution of a sensing action [Scherl & Levesque, 1993], [Levesque, 1996]. Formal accounts based on this view concentrate on the difficulties associated with formalising the effects of knowledge producing actions, and the task of planning with such actions. According to the second view, perception is a passive process, whereby the robot's model of the world is updated as a side-effect of its physical actions [Shanahan, 1996], [Shanahan, 1997]. In formal accounts based on this view, abduction is used to supply possible explanations of incoming sensor data, and the results of this abductive process are assimilated into the robot's representations.

In both approaches to perception, the flow of information is one-way — from raw sensor data to high-level representations. By contrast, it has long been recognised that in biological brains, the flow of information between low-level perception and high-level cognition is bidirectional [Cavanagh, 1999]. One of the most prominent mechanisms for achieving this bidirectionality is *expectation*. Low-level perceptual cues suggest interpretations of raw sensor data that line-up with past experience. This leads to the expectation of certain features in the environment, which in turn feeds back to low-level perceptual systems, making them more sensitive to those features. Initial expectations are then either confirmed or disconfirmed, and a stable model of the environment is built up.

From an engineer's point-of-view, it makes good computational sense to adopt this biologically inspired technique. Take vision, for example, which is the main subject of the present paper. A robot's visual system delivers a large amount of raw sensor data. In the unidirectional approach, this mass of sensor data is processed using a battery of methods for extracting features, such as edges and patches of uniform texture, which are then used to segment the image, picking out the foreground objects. This is a highly computationally intensive business. In a cluttered scene, a large number of features will be thrown up, which need to be sifted and sorted. Moreover, the process is highly sensitive to noise. Shadows, poor lighting, highlights, and surface patterns can all render the output of the system useless.

The benefits of incorporating a two-way flow of information include a reduction in computation and increased robustness. A reduction in computation is achieved because the sensitivity of low-level image processing routines can be initially set low, so that only a small number of highly prominent features are passed up to the next highest level of processing. Prominent features act as cues leading to the expectation of other features in the visual field. These expectations are fed back down to the low-level perceptual system, resulting in a selective increase of sensitivity confined only to those areas of the visual field that are potentially interesting. Increased robustness also results, first because of the initial insensitivity of the low-level processing, and second because the feedback loop soon

eliminates from consideration features leading to expectations that fail to be confirmed.

Similar arguments to these were used to motivate pioneering work in *active vision* [Aloimonos, *et al.*, 1987], [Ballard, 1991]. Research on active vision emphasises, among other things, the computational benefits of selective head or camera movements to fix on portions of the scene of interest while filtering out the rest. In a sense, the rationale behind the present paper is the same. The chief differences are that the active component of the system presented here is not at the level of physical movement, but at the level of software adjustments to low-level image processing routines. However, in more general terms, the paradigm offered here encompasses a spectrum of possibilities, all involving the following three steps of hypothetico-deductive reasoning.

1. Generate competing hypotheses to explain the sensor data.
2. Determine the consequences (expectations) of those hypotheses.
3. Carry out actions that will confirm or disconfirm these expectations, thus ruling out one or more of the competing hypotheses.

A confirming/disconfirming action might take several forms. At one end of the spectrum, it could be the software adjustment of a parameter of a low-level image processing algorithm, such as edge detection. At the other end of the spectrum, it could be a knowledge producing action, or knowledge producing plan, such as "Find a telephone directory and look up John's number" [Scherl & Levesque, 1993]. Traditional active perception, in which the actions in question are small-scale physical adjustments of the sensory apparatus, can be thought of as lying somewhere in the middle of this spectrum. Although the examples presented here are all of the first type, the theoretical treatment is intended to be generic.

Specifically, this paper presents a formal, logic-based account of visual perception incorporating a two-way flow of information between low-level image processing routines and high-level reasoning. The formalisation is a modification of the abductive model of perception presented in [Shanahan, 1996]. In this modified account, feedback and expectation are incorporated via the preference relation that selects between competing hypotheses. To facilitate this, two measures of the explanatory value of a hypothesis are proposed and compared — an *ad hoc* measure based on some obvious design criteria, and a more principled probabilistic measure. This general account is applied to a specific vision task, namely the recognition of cuboidal objects in a cluttered, noisy scene.

2 VISUAL PERCEPTION AS ABDUCTION

According to the model of robot perception put forward in [Shanahan, 1996], the task of robot perception is characterised roughly as follows. Given a stream of sensor data represented by the conjunction Γ of a set of observation sentences, find one or more explanations of Γ in the form of a logical description Δ of the locations and shapes of hypothesised objects, such that,

$$\Sigma \wedge \Delta \models \Gamma$$

where Σ is a background theory describing how objects in the world impact on the robot's sensors. The form of Δ, that is to say the terms in which an explanation must be couched, is prescribed by the domain. Typically there are many Δs of the permitted form that might explain a given Γ according to this definition. So a preference relation is defined for ordering them. This preference relation will be the vehicle for introducing expectation and feedback in the next section.

While this basic form of abduction is adequate for very rudimentary forms of robot perception — with bump switches or infra-red proximity sensors, for example [Shanahan, 1996] — its limitations soon become apparent when we try to apply it to a richer sensory modality like vision. To begin with, we need to relax the constraint, implicit in the above definition, that an explanation must be found for *all* the sensor data, that is to say for the whole of Γ. Snapshots of the robot's visual field contain a large number of features, and it would be inappropriate for the perceptual process to try to build a model of the world that accounts for every one of them. On the contrary, we want a perceptual system to ignore irrelevant data and pick out only those objects in the scene that are pertinent to the robot's goals or desired behaviour.

More important still, in the context of the present paper, is the fact that the basic abductive account leaves no room for the sort of two-way flow of information whose benefits were outlined in the introduction. It's especially galling that basic abduction has no means for allowing high-level declaratively represented expectations to inform low-level perceptual mechanisms when we consider that one of the chief attractions of an abductive treatment is that it is expressed in high-level, declarative terms. (For another attempt to reconcile abduction with top-down perceptual processing, see [Josephson & Josephson, 1994], Chapter 10.)

To see how this situation might be remedied, let's consider a motivating example. The image at the top of Figure 1 is the output from one of the stereoscopic cameras mounted on the head of an upper-torso humanoid robot. The image at the bottom shows the result of applying a standard edge detection algorithm, using the Sobel operator, with a high threshold. Let's

look more closely at the block circled in white in the unprocessed image. Our aim here is to devise a mechanism that will exploit the cues present in the image to conclude that a block of the right shape and size is out there, and which won't be distracted by the fact that the block is composed of two differently coloured halves, one of which is almost lost in the shadows.

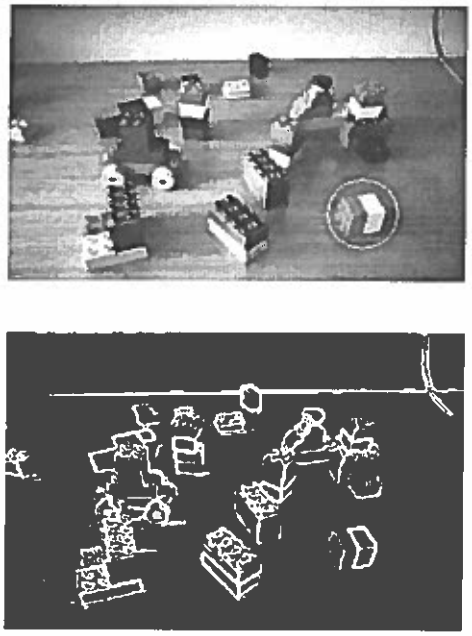


Figure 1: Some Raw Visual Data

More precisely, the problem is this. When a standard region finding algorithm is applied to the edge data in Figure 1, it comes up with the set of lines shown in bold in Figure 2, namely AB, AD, BC, DE, CF, and EF. Other lines, such as GC and HE are visible to the human eye, but the region finding algorithm is blind to them. If we turn up the sensitivity of the edge detector sufficiently to make these lines visible to the region finder, the number of spurious, unwanted lines thrown up as well — caused by the grain of the wooden table-top, for example — is so large that it becomes computationally infeasible to attempt to interpret them all, sorting out the useful data from the rest.

On the basis of this meagre and misleading data, how are we to find the true outline of the block? An abductive approach to perception initially seems like a good idea here. The correct hypothesis — let's call it H_1 — that the visible lines are caused by a block whose base is the line HF, is indeed sanctioned by the abductive definition. But so is the alternative hypothesis — which we'll call H_2 — that the visible lines are caused by a block whose base is the line EF. Moreover, according to almost any plausible preference relation we can think of for ordering multiple explanations, the second, incorrect hypothesis comes out on top, due to the influence of the line AD.

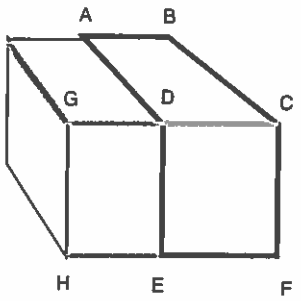


Figure 2: An Ambiguous Block

The main innovation of this paper is to supply a modified abductive treatment of perception in which the expectations generated by each hypothesis feed back into the preference ordering for choosing between them. In the present example, hypothesis H_1 generates the expectation (among others) that the line DC extends beyond D. Hypothesis H_2, on the other hand, generates the expectation of a vertical line extending downwards from A. These expectations retrospectively adjust the preference ordering for multiple explanations at the next level down in the hierarchy of sensor data processing, namely the process of finding edges and lines. This translates into a highly selective increase in the sensitivity of the edge detector at precisely those places where edges are expected. With this increase in sensitivity, the expectations of hypothesis H_1 are confirmed, and those of hypothesis H_2 are disconfirmed, leading to a revised ordering of the hypotheses in which H_1 comes out on top.

3 EXPLANATORY VALUE AND EXPECTATION

In this section, a formal description of abductive perception with feedback is presented. The language and consequence relation of standard first-order predicate calculus are used throughout. First, we have a basic definition of explanation, inherited from previous work on the topic of abductive perception [Shanahan, 1996]. A background theory Σ, a set of atomic *abducible* formulae, and a set of atomic *explainable* formulae are assumed. A *data set* is a conjunction of explainable formulae.

Definition 3.1. Given a background theory Σ, a conjunction ϕ of abducible formulae is an *explanation* for a data set Γ if,

$$\Sigma \wedge \phi \models \Gamma$$

and $\Sigma \wedge \phi$ is consistent. □

In general, the perceptual system of a robot will be many layered. That is to say, low-level features such as pixels will be interpreted in terms of higher-level features such as lines, which in turn will be interpreted in terms of surfaces, which are finally interpreted in terms of solid shapes. However, in the formal account that follows, these will be compressed into a single layer. The generalisation to multiple layers is straightforward. (See [Josephson & Josephson, 1994], Chapter 10 for an abductive account of layered perception.)

In the following account, we suppose the presence of a mass of sensor data. The focus here is visual perception, and the mass of data in question is taken to be the set of edges detected in a single snapshot of the visual field. Dynamic aspects of the data are neglected in the present paper, but obviously motion cues are a valuable source of additional information.

Some form of attention mechanism is taken for granted in the scheme described below. In effect, this means that a small portion of the visual field is isolated, such as the circle in Figure 1, and the task is to find the best explanation for the edges within that area. (This is obviously analogous to the visual attention mechanism of the human brain, whereby eye saccades and head movements cause the eye's foveal area to alight on objects of interest.) If the sensor data of interest is described by a data set Γ, the challenge is to determine the set of potential explanations for Γ, and to order them.

A variety of ways to order alternative explanations of the same sensor data might be employed. To facilitate the introduction of expectation and feedback, the method of choice here will be to assign a numerical score, in the range 0 to 1, to each hypothesised feature that explains Γ. This score is intended to reflect the *explanatory value* of the hypothesis. One measure of explanatory value is proposed in this section, and an alternative based on probability is put forward in Section 6. The set of features is partitioned into *primitive* and *non-primitive*, where primitive features are the raw sensor data themselves, and a non-primitive feature is an abducible formula, representing, for example, the presence of a block in a certain location. The initial explanatory value of a primitive feature is a function of the raw sensor data, scaled to fit the range 0 to 1. The initial explanatory value of a non-primitive feature is a weighted average of the explanatory values of those features it explains.

Definition 3.2. The *initial explanatory value* $V_0(H)$ of a non-primitive feature H is equal to,

$$\left(1-\frac{1}{n+1}\right)\left(\frac{1}{n}\sum_{\psi\in\Psi}V_0(\psi)\right)$$

where,

- Ψ is the set of all ψ such that H is an explanation for ψ and $V_0(\psi) > \theta_1$,
- n is the cardinality of Ψ. □

The idea of the weighting factor is to give the highest value to the feature that explains the most lower-level features. However, the effect of this weighting diminishes as the number of explained features goes up. The role of the threshold value θ_1 is crucial. Only those lower-level features that themselves have a high enough explanatory value are worth explaining. In terms of the example of the previous section, pixels with too low a Sobel value are ignored.

Now we introduce expectation and feedback, which is the means by which those features too faint to be considered initially get taken into account without swamping the system with computational demands. The initial explanatory value of a feature is increased to the extent that it fulfills the expectations of a higher-level feature, and is reduced to the extent that its own expectations fail to be fulfilled.

Definition 3.3. Let k be any integer greater than zero, and H be any feature. The *unmoderated* k^{th} *explanatory value* $V_k^-(H)$ of λ is equal to,

$$V_{k-1}(H) - k.Q\left(1-\frac{1}{m+1}\right)\left(\frac{1}{m}\sum_{\phi\in\Phi}V_{k-1}(\phi)\right) + \omega$$

where

- Φ is the set of all ϕ such that H is an explanation for ϕ and $V_{k-1}(\phi) \le \theta_2$,
- m is the cardinality of Φ, and
- if
  - $V_{k-1}(H) > \theta_2$, and
  - there exists an α which is an explanation for H such that $V_{k-1}(\alpha) > \theta_1$.

  then $\omega = R$, else $\omega = 0$. □

The constant Q dictates the degree to which an unfulfilled expectation reduces a feature's explanatory value. Similarly, the constant R dictates the degree to which a feature's explanatory value is enhanced if it does fulfill some expectation. The constant θ_2 represents the threshold of explanatory value below which a feature is considered to be effectively absent and therefore to disconfirm higher-level features that expect it. The behaviour of algorithms based on the definitions given here is quite sensitive to the values of these constants, as discussed below.

Definition 3.3 can yield values greater than 1 or less than 0. The following definition imposes the corresponding ceiling and floor.

Definition 3.4. The k^{th} *explanatory value* $V_k(H)$ of a non-primitive feature H is defined as follows.

$$V_k(H) = \begin{cases} V_k^-(H) \text{ if } 0 \le V_k^-(H) \le 1 \\ 1 \text{ if } V_k^-(H) > 1 \\ 0 \text{ if } V_k^-(H) < 0 \end{cases} \quad \square$$

The repeated application of feedback generally takes the explanatory value of each feature to a fixpoint. In dynamical systems terms, this is an attractor basin, analogous to the attractor basins in, for example, the state space of a Hopfield net that might similarly be used to recognise patterns in sensor data on the basis of expectation and past experience. In contrast to a neural net, however, the present system enjoys the advantages of being able to carry out symbolic reasoning with declaratively represented knowledge.

Definition 3.5. The *final explanatory value* $V_\infty(H)$ of a non-primitive feature H is equal to $V_m(H)$ where m is the smallest integer such that,

$$V_m(H) = V_{m+1}(H) \quad \square$$

Further investigation is required into the conditions under which there exists a final explanatory value for a feature. However, none of the (limited) experiments carried out so far has unearthed an unstable combination of logical formulae and numerical values.

Definition 3.6. The *explanatory value* of a set of features is the mean explanatory value of the set. $\square$

Now, the preference relation used to order competing explanations for the same item of sensor data mirrors the assignment of final explanatory values to those explanations.

4 EXPECTATION AND FEEDBACK IN ACTION

The definitions above tell us very little about what conditions that cause a feature's explanatory value to ascend to a fixpoint and what conditions cause it to descend to a fixpoint. Further investigation of this issue is required, but initial experimentation has shown that a vision algorithm based on the definitions of the previous section is effective — in the sense that the explanatory values move in the required directions in response to feedback — when θ_1=0.4, $\theta_2 = \frac{2}{3}\theta_1$, R=0.1, and Q=1.0.

The following example illustrates the way expectation and feedback can distinguish between competing hypotheses. The parameters θ_1, θ_2, R, and Q are set to the above values. Suppose a snapshot of a robot's sensor data includes the primitive features F_1 to F_6, with the following initial explanatory values.

$V_0(F_1) = 0.36$ $\quad$ $V_0(F_2) = 0.63$
$V_0(F_3) = 0.81$ $\quad$ $V_0(F_4) = 0.72$
$V_0(F_5) = 0.81$ $\quad$ $V_0(F_6) = 0.20$

Now suppose we have two non-primitive features H_1 and H_2 such that,

$\Sigma \models \neg [H_1 \wedge H_2]$
$\Sigma \wedge H_1 \models F_1 \wedge F_2 \wedge F_3 \wedge F_4$
$\Sigma \wedge H_2 \models F_3 \wedge F_4 \wedge F_5 \wedge F_6$.

In other words, H_1 and H_2 are competing hypotheses, each of which explains some features of the sensor data. However, the initial explanatory value of H_2 is slightly higher than that of H_1.

$V_0(H_1) = 0.54$
$V_0(H_2) = 0.58$

So, at first glance, H_2 looks a better bet than H_1. However, the two hypotheses have different expectations. In particular, H_1 leads to the expectation of F_1, while H_2 gives rise to the expectation of F_6. Neither F_1 nor F_6 were prominent enough to contribute to the initial explanatory values of the hypotheses. But with the application of feedback, these differing expectations start to make a difference, and the two hypotheses swap places. In particular, we have,

$V_1(H_1) = 0.58$
$V_1(H_2) = 0.56$

From this point on, the two hypotheses start to diverge. This divergence reflects the fact that the initial explanatory value of F_1 is sufficient to fulfill the expectation of H_1, while the initial explanatory value of F_6 is so low that it disconfirms H_2. The final explanatory values of the hypotheses end up at opposite ends of the spectrum.

$V_\infty(H_1) = 0.80$
$V_\infty(H_2) = 0.00$

It might seem counter-intuitive that the final explanatory value of H_2 is zero. After all, this hypothesis surely has some value. In effect, this outcome is an artefact of the way thresholds are used. The result of applying feedback is to amplify the effects of confirmation and disconfirmation until certain hypotheses are dismissed altogether. The method of applying feedback with the probabilistic measure of explanatory value presented in Sections 6 and 7 is less polarising.

This is obviously a simple example, and with a more complex network of logical relations, the dynamics of the system is correspondingly more complicated. However, it should be clear that the example maps easily to the benchmark vision problem from Section 2.

The next section offers a more formal treatment of that benchmark problem.

5 VISION REVISITED

In this section, we take a closer look at the benchmark vision problem of Section 2. The main task is to formalise it as an abductive explanation problem, to which the definitions of the previous section are applicable. To this end, a series of axioms is presented describing blocks in terms of surfaces, surfaces in terms of lines, and the appearance of lines in terms of visible edges.

The first axiom says that, under the right conditions, a linear feature in 3D space gives rise to a visible edge (according to the Sobel operator) in the robot's visual field. The linear feature might correspond to a spatial discontinuity, such as the side of an object, or to a colour discontinuity, such as a shadow across the surface of an object. This axiom is used to reason abductively from data items in the robot's two-dimensional visual field (edges) to spatially-located features of the robot's workspace.

$$[Line(w) \wedge FromTo(w,p1,p2) \wedge \neg Occluded(w) \wedge p3=Project(p1) \wedge p4=Project(p2)] \rightarrow \exists e\, [Edge(e) \wedge FromTo(e,p3,p4)] \quad (1)$$

The formula Line(w) represents that a linear feature w exists in 3D space. The term Project(p) denotes the point in the robot's visual field onto which point p in 3D space projects. The formula Edge(e) represents the presence of an edge e in the robot's visual field. The formula FromTo(x,p1,p2) indicates the end points, p1 and p2, of the line or edge x. These will be 3D co-ordinates in the case of a line (in the robot's workspace), and 2D co-ordinates in the case of an edge (in the robot's visual field).

The formula Occluded(w) holds if some object lies between w and the robot's viewpoint. Several other kinds of exception, in addition to occlusion, would merit inclusion in a fuller axiomatisation, such as poor lighting conditions, a faulty camera, and so on. In Sections 6 and 7, there will be a requirement for explicit noise and abnormality terms in formulae like Axiom (1).

The next two axioms are used to reason abductively from lines to surfaces and their properties. The two axioms correspond to the two possible explanations of a linear discontinuity — either it's the edge of a solid object or it's a surface feature such as a shadow. The formula Bounded(s,w1,...,w4) means that the rectangular surface s is bounded by the four lines w1 to w4. The formula Marked(s,w) means the surface s has a linear feature w.

$$[Surface(s) \wedge Bounded(s,w1,w2,w3,w4)] \rightarrow [Line(w1) \wedge Line(w2) \wedge Line(w3) \wedge Line(w4) \wedge Parallel(w1,w2) \wedge Parallel(w3,w4)] \quad (2)$$

$$[Surface(s) \wedge Marked(s,w)] \rightarrow Line(w) \quad (3)$$

Finally, Axiom(4) below takes the abductive process from surfaces to cuboidal blocks. The formula Faces(b,s1,...,s6) means that the block b has the six faces s1 to s6, each of which is a rectangular surface.

$$[Block(b) \wedge Faces(b,s1,s2,s3,s4,s5,s6)] \rightarrow [Surface(s1) \wedge Surface(s2) \wedge Surface(s3) \wedge Surface(s4) \wedge Surface(s5) \wedge Surface(s6) \wedge PParallel(s1,s2) \wedge PParallel(s3,s4) \wedge PParallel(s5,s6)] \quad (4)$$

These and similar axioms form the basis of a visual perception system that combines high-level logical reasoning with feedback and expectation to interpret scenes that would present a challenge to a conventional image understanding system. The knowledge inherent in Axioms (1) to (4) is analogous to a 3D model in a conventional machine vision system, and the abductive interpretation of the sensor data corresponds to the conventional process of matching a model to the data from a given scene [Jain, *et al.*, 1995, Chapter 15]. However, under the present scheme, it's possible to augment this knowledge with declarative sentences, such as "all the red blocks are shorter than all the blue blocks" or "one of the green blocks is hidden". Combined with a suitable inference mechanism, such knowledge can be used to influence low-level processing.

To see how the abductive process works, let's return to the simple example of Section 2. Suppose we have the following set Γ of items of low-level sensor data (Figure 3).

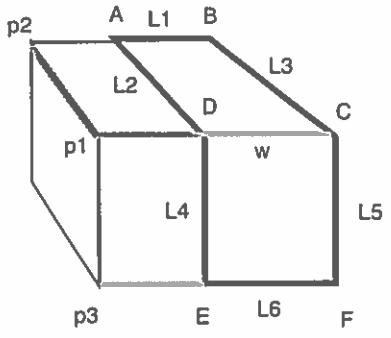


Figure 3: Hypothesised Lines and Points

$Edge(L_1) \wedge FromTo(L_1,A,B)$
$Edge(L_2) \wedge FromTo(L_2,A,D)$
$Edge(L_3) \wedge FromTo(L_3,B,C)$

$\text{Edge}(L_4) \wedge \text{FromTo}(L_4,D,E)$

$\text{Edge}(L_5) \wedge \text{FromTo}(L_5,C,F)$

$\text{Edge}(L_6) \wedge \text{FromTo}(L_6,E,F)$

Let the set of abducible formulae be all those of the form Block(b), Faces(b,s1,...,s6), or Marked(s,w). Now, let H_1 be,

$\text{Block}(B_1) \wedge \text{Faces}(B_1,S_{1,1},\ldots,S_{1,6})$

where, for some $S_{1,i}$ and $S_{1,j}$ in $\{S_{1,1},\ldots,S_{1,6}\}$, we have,

$\text{Bounded}(S_{1,i},L1,L2,L3,w)$

$\text{Bounded}(S_{1,j},w,L5,L6,L4)$

for some hypothesised line w. Now, let H_2 be,

$\text{Block}(B_2) \wedge \text{Faces}(B_{2,1},S_{2,1},\ldots,S_{2,6})$

where, for some $S_{2,i}$ and $S_{2,j}$ in $\{S_{2,1},\ldots,S_{2,6}\}$ we have,

$\text{Bounded}(S_{2,i},L_3,w_1,w_2,w_3)$

$\text{Bounded}(S_{2,j},L_5,w_1,w_4,w_5)$

for hypothesised lines $w_1,\ldots,w_5$, such that,

$\text{FromTo}(w_1,C,p_1)$ $\text{FromTo}(w_2,p_1,p_2)$

$\text{FromTo}(w_3,p_2,B)$ $\text{FromTo}(w_4,F,p_3)$

$\text{FromTo}(w_5,p_3,p_1)$

for hypothesised points p_1 to p_3, all distinct from points A to F. Next, let H_3 be,

$\text{Marked}(S_{2,i},L_2)$.

Finally, let H_4 be,

$\text{Marked}(S_{2,j},L_4)$.

It can be straightforwardly shown that H_1 is an explanation for Γ, as is the conjunction of hypotheses H_2 to H_4. The latter combination is, of course, correct, while H_1 is incorrect. However, when data from the actual vision system is plugged in, the initial explanatory value of H_1 is much higher than that of the correct combination. With the application of expectation and feedback, using the definitions of Section 3, this situation is reversed. The incorrect hypothesis H_1 leads, through Axiom (4), to the expectation of a vertical surface, one of whose edges is AD. The failure of this expectation brings about a fall in the explanatory value of H_1. On the other hand, the initially unpromising hypothesis H_2 leads to the expectation of a line extending from B to A and beyond, to a point p_2. This expectation is confirmed, causing the explanatory value of H_2 to rise. In the end, the final explanatory value of the correct combination of hypotheses exceeds that of the incorrect hypothesis, which is the desired result.

6 A PROBABILISTIC MEASURE OF EXPLANATORY VALUE

The measure of explanatory value proposed in Section 3 forms the basis of a workable system for perception incorporating expectation and feedback. However, from a theoretical perspective, the proposal looks *ad hoc*, because the numerical values defined lack any sort of semantic justification. In this section, an alternative measure of explanatory value is devised, based on probability. As we'll see, when compared to the more *ad hoc* measure, the probabilistic version has both advantages and disadvantages.

Suppose we have a set F of abducible formulae. In the absence of further information, all formulae in F are assumed to be independent and to have equal prior probability p. Note that, in the absence of further information, the prior probability $P(H)$ of any conjunction $H = f_1 \wedge \ldots \wedge f_n$ of abducible formulae is p^n.

Now suppose we acquire some sensor data, represented as a formula Γ. Using abduction, we find the set of possible explanations of Γ is $\{H_1 \ldots H_m\}$, where each H_i is a conjunction of abducible formulae. Suppose the set of hypotheses is mutually exclusive. So we have,

$H_1 \oplus \ldots \oplus H_m$.

Using basic probability theory, we can now assign an explanatory value to each candidate explanation H_i, corresponding to the posterior probability that it is true, given the above disjunction. From Bayes' Theorem we have,

$$P(q \mid q \oplus r) = \frac{P(q \oplus r \mid q).P(q)}{P(q \oplus r)}$$

which simplifies to,

$$P(q \mid q \oplus r) = \frac{P(q)}{P(q) + P(r)}.$$

Now, by setting q equal to any hypothesis H_i and r to the rest of the hypotheses, we can calculate the posterior probability of H_i given Γ, under the assumption that $H_1 \ldots H_m$ is the set of all the possible explanations of Γ. A related formula is derived by Poole [1993] for abduction with probability. Poole's equation yields the conditional probability $P(q|\Gamma)$, and is equivalent to the present formula under the same assumption.

Now suppose Γ comprises n separate items of sensor data, each requiring explanation. In general, a hypothesised object will only explain a portion of the sensor data of interest. So a complete explanation will comprise a number of hypothesised objects supplemented with a number of *noise terms* to "explain away" the rest of the sensor data [Poole, 1995], [Shanahan, 1997].

To allow for noise terms, the background theory Σ must be supplemented with some additional formulae. For example, as well as Axiom (1) from Section 5, we might have,

[Noise(w) ∧ FromTo(w,p1,p2) ∧
p3=Project(p1) ∧ p4=Project(p2)] →
∃ e [Edge(e) ∧ FromTo(e,p3,p4)].

Let the set F of abducibles comprise a set F_a of noise terms each with probability p_a and a set F_b of propositions positing spatially-located objects, each with probability p_b. Formulae in F_b explain multiple items of sensor data while formulae in F_a explain only 1 item of sensor data. What we're interested in is how the probability (and thus the explanatory value) of a hypothesis grows with the number of items of sensor data it explains.

Suppose the disjoined set of explanations of Γ obtained by abduction is $H_1 \oplus ... \oplus H_m$, such that,

$$H_1 = [f \wedge f_{k+1} \wedge ... \wedge f_n]$$

where f is drawn from F_b and each f_i is drawn from F_a. In other words, H_1 explains k out of n items of sensor data, and the rest have to be explained by noise terms. We have,

$$P(H_1) = p_b p_a^{n-k}$$

Let R be $H_2 \oplus ... \oplus H_m$. Then we have,

$$P(H_1 \mid H_1 \oplus R) = \frac{p_b \cdot p_a^{n-k}}{p_b \cdot p_a^{n-k} + P(R)}$$

or alternatively,

$$P(H_1 \mid H_1 \oplus R) = \left(1 + \frac{P(R)}{p_b p_a^{n-k}}\right)^{-1}$$

Note that the posterior probabilities of H_1 to H_m sum to 1, as we would expect. To simplify the analysis, let's assume from now on that each hypothesis posits exactly one object. So each data item unaccounted for by that object is explained away with a noise term. Plugging in the appropriate formula for $P(R)$, this leads to the following probabilistic measure of initial explanatory value, which is the counterpart to Definition 3.2. The set of data items to be explained will be those in the area of interest whose value exceeds some threshold. We'll designate this threshold θ_1, since it corresponds to θ_1 in Section 3.

Definition 6.1. Given a data set Γ, the *probabilistic explanatory value* $VP(H,\Gamma)$ of a hypothesis H is equal to,

$$\left(1 + \frac{\sum_{\phi \in \Phi} p_a^{S_\phi}}{p_a^{S_H}}\right)^{-1}$$

where

- Φ is the set of all hypotheses that explain Γ, and
- S_ϕ is the number of terms in hypothesis ϕ drawn from F_a. □

Note that the p_b term always drops out. (This is only the case if each hypothesis posits the same number of objects.) Let's apply this definition to the example of Section 4. Let θ_1 be 0.4. Since both hypotheses explain 3 out of 4 items of sensor data, regardless of the value of p_a, we get,

$$VP(H_1,\Gamma) = VP(H_2,\Gamma) = 0.5.$$

Let's pick a different example. Suppose the set of explanations is $\{H_1,H_2\}$, and that H_1 and H_2 respectively explain 1 out of 4 and 2 out of 4 data items without recourse to noise terms. In other words, H_1 includes 3 noise terms while H_2 includes 2 noise terms. Plugging these values into the formula, we get,

$$VP(H_1,\Gamma) = \left(1 + \frac{1}{p_a}\right)^{-1}$$

and,

$$VP(H_2,\Gamma) = (1 + p_a)^{-1}.$$

So, for example, if we let p_a=0.1, the posterior probability (explanatory value) of H_1 is = 0.09 and the posterior probability of H_2 is 0.91 On the other hand, if we let p_a=0.5, the respective values we get for H_1 and H_2 are 0.33 and 0.67. These examples show that the probabilistic measure of explanatory value does assign higher values to hypotheses that explain more data. So it meets the most important criterion for a measure of explanatory value. But by inspecting the two formulae (the probabilistic version and the more *ad hoc* formula in Definition 3.2), we also see the many differences between the two measures.

- The probabilistic measure takes account of the number of unexplained data items rather than the proportion of data items explained. So, all other things being equal, it assigns the same explanatory value to a hypothesis that explains 99 out of 100 data items as to a hypothesis that explains 1 out of 2. In this respect the *ad hoc* measure has more appeal.
- The probabilistic measure, unlike the *ad hoc* measure, isn't weighted by the value of the sensor data explained. So it doesn't distinguish hypotheses that explain prominent sensor data

from those that explain poor quality sensor data. The assumption behind the probabilistic measure is that all the data items in Γ are present and all require explanation.

- The probabilistic measure takes account of the whole set of competing hypotheses when assigning an explanatory value to each member of that set. The *ad hoc* measure assigns the same value to a hypothesis that is a unique explanation of the data as to a hypothesis that is one among ten competitors. In this respect, the probabilistic measure is to be preferred.
- The probabilistic measure takes into account the probability of a noise term. According to the probabilistic measure, the more noisy the data, the narrower the gap between a hypothesised object that explains a lot and one that explains only a little. This makes sense, so in this respect again the probabilistic measure wins out.

7 ABDUCTION, PROBABILITY, AND FEEDBACK

The question now, of course, is how to incorporate the probabilistic measure of explanatory value into a scheme that exploits expectation and feedback. With the *ad hoc* measure, the explanatory value of a hypothesis is increased if it fulfills the expectation of another hypothesis, and decreased if its own expectations are unfulfilled (Definition 3.3). With the probabilistic version, a different strategy will be adopted, whereby the set Γ of data items to be explained is increased, as a result of feedback, to include the outcome of testing the expectations of all the candidate hypotheses. The tricky part is to augment Γ in such a way as to include the *absence* of expected data items as well as their presence.

First, the notion of a data set is widened to include negated formulae. These will be used to represent unfulfilled expectations.

Definition 7.1. An *augmented data set* is a conjunction of the form $(\neg)\psi_1 \wedge \ldots \wedge (\neg)\psi_n$, where each ψ_i is a data item. □

The definition of an explanation now has to be modified to cater for augmented data sets. The important thing to note here is that there is no requirement for an explanatory hypothesis to entail a negated formula in the data set, but simply to be consistent with it.

Definition 7.2. Let Γ be an augmented data set. Given a background theory Σ, a conjunction ϕ of atomic abducible formulae is an *explanation* of Γ if, for all positive formulae ψ in Γ,

$$\Sigma \wedge \phi \models \psi$$

and, for all negative formulae $\neg\psi$ in Γ,

$$\Sigma \wedge \phi \not\models \psi$$

and $\Sigma \wedge \phi$ is consistent. □

For this definition of explanation to be effective, non-monotonicity needs to be introduced. Specifically, the expectations of a hypothesis must be given the status of defaults. In the same way that a noise term can be introduced to fill in gaps in the explanation, an abnormality term can be introduced to override a default expectation that was unfulfilled. But when it comes to calculating the explanatory value of a hypothesis, the inclusion of an abnormality term, like the inclusion of a noise term, is costly, as we'll see shortly.

Many of the formulae in the background theory Σ need to be rewritten to turn them into default rules. For example, Axiom (1) from Section 5 becomes,

$$[\text{Line}(w) \wedge \text{FromTo}(w,p1,p2) \wedge \neg \text{Occluded}(w) \wedge \neg \text{Abnormal}(w) \wedge p3=\text{Project}(p1) \wedge p4=\text{Project}(p2)] \rightarrow \exists e\, [\text{Edge}(e) \wedge \text{FromTo}(e,p3,p4)].$$

Default reasoning can now be effected through circumscription. Each hypothesis is circumscribed, minimising the Abnormal predicate. No further details will be given here, and from now on, all hypotheses should be assumed to be implicitly circumscribed in this way. Because only atomic abnormality formulae are made abducible, the mathematics of this is straightforward, always yielding simply the completion of the Abnormal predicate. In real-world terms, the abnormality predicate represents a whole raft of circumstances that might explain the non-appearance of an expected edge, such as poor lighting, lack of contrast between foreground and background, occluding objects, and so on.

The set F of abducible formulae has to be expanded to include abnormality terms. In particular, let's assume the subset F_a of F now comprises both noise terms and abnormality terms, each with prior probability p_a.

We're now in a position to define an augmentation operator that adds both fulfilled and unfulfilled expectations to a data set. First we make precise the notion of an expectation.

Definition 7.3. Given a background theory Σ, the set of expectations $Exp(H)$ of a hypothesis H comprises all data items ψ such that,

$$\Sigma \wedge H \models \psi$$ □

Now we define the augmentation operator, and introduce two new threshold values, θ_2 and θ_3 (Figure 4). Intuitively, the meaning of these thresholds is as follows. If the value assigned to a data item is above θ_1, then that data item is assumed to be present, as before.

If it falls below θ_3, then it's assumed to be absent. And if it falls between θ_2 and θ_1 then it's assumed to present if it was expected.

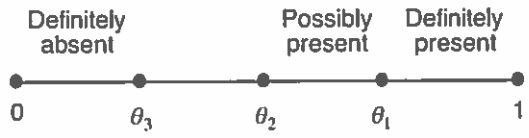


Figure 4: Data Item Thresholds

Definition 7.4. If Γ is a data set, then the augmented data set $Aug(\Gamma)$ is $\Gamma \cup P \cup N$, where,

- P is the set of all ψ such that $\psi \in Exp(H)$ for some H where $VP(H,\Gamma) \geq \theta_1$, and the value of $\psi \geq \theta_2$, and
- N is the set of all formulae of the form $\neg\psi$ such that $\psi \in Exp(H)$ for some H where $VP(H,\Gamma) \geq \theta_1$, and the value of $\psi < \theta_3$. □

In the example of Section 4, if θ_1=0.4, θ_2=0.35, and θ_3=0.30, then $Aug(\Gamma) = \{F_1, F_2, F_3, F_4, F_5, \neg F_6\}$. So H_1 contains only 1 noise term (for F_5) to explain 6 data items, while H_2 contains 2 noise terms (to account for F_1 and F_2) and one abnormality term (to account for $\neg F_6$). If we let p_a=0.1, this gives

$VP(H_1, Aug(\Gamma))$=0.99

$VP(H_2, Aug(\Gamma))$=0.01.

More realistically, if we let p_a=0.5, we get,

$VP(H_1, Aug(\Gamma))$=0.8

$VP(H_2, Aug(\Gamma))$=0.2.

Either way, hypothesis H_1 emerges as the clear winner, as we would expect. Note that $Aug(Aug(\Gamma))=Aug(\Gamma)$. In other words, further applications of the Aug operator have no effect in this case.

8 CONCLUDING REMARKS

The methods described above have been incorporated in an implemented vision system for deployment on an upper-torso humanoid robot which is currently under construction at Imperial College (Figure 5). The robot has two arms, each with three degrees-of-freedom, and a pan-and-tilt head with stereoscopic vision. (The exploitation of depth information from the stereo cameras is one of many topics for further investigation.) The robot's forearms are mechanically constrained to be parallel to the workbench in order to cut down on degrees-of-freedom and simplify the control issues. In its first incarnation, the robot will be equipped with simple prods, not grippers or hands, at the ends of its forearms.

The robot's task is to survey a table-top cluttered with objects. Most of the objects will be unfamiliar, but some will be cuboidal. The robot's perceptual system needs to pick out these objects, and form a representation of their locations and dimensions. The robot, or a demonstrator, will nudge the objects, and the robot must track familiar objects, increasing the quality of its representations as the objects are viewed from new angles and previously unseen parts become visible. Initial experiments using the vision system described are promising.

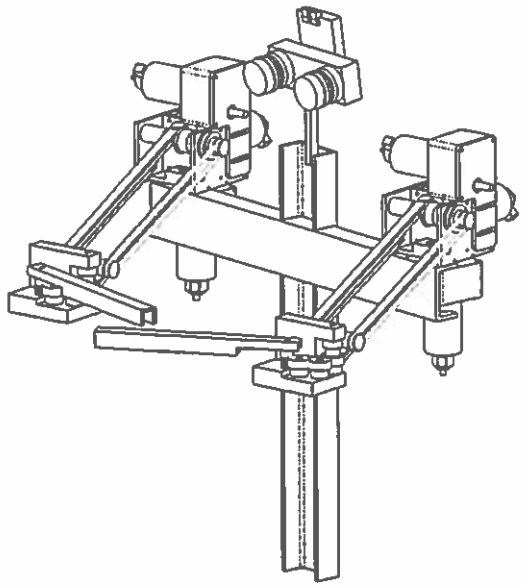


Figure 5: LUDWIG the Humanoid Robot

Several avenues of further research are being followed. These include the use of more sophisticated forms of high-level reasoning in the perceptual process, in order to deal with occlusion and object persistence [Randell, *et al.*, 2001]. This necessitates taking account of the ongoing actions of the robot and other agents [Shanahan, 2000], ensuring that, as in work on animate vision, the dynamics of the robot's interaction with the world enhance, rather than handicap, the perceptual capabilities of the robot.

Acknowledgements

This work was carried out as part of EPSRC grant GR/N13104 "Cognitive Robotics II". Thanks to Yiannis Demiris, Seçil Ozen, Dave Randell, Paulo Santos, Ray Thompson, and Mark Witkowski. Thanks to the anonymous referees for their insight and generosity.

References

[Aloimonos, *et al.*, 1987] J.Aloimonos, I.Weiss, and A.Bandyopadhyay, Active Vision, *Proceedings 1st International Conference on Computer Vision*, pp. 35–54.

[Ballard, 1991] D.H.Ballard, Animate Vision, *Artificial Intelligence*, vol. 48 (1991), pp. 57–86.

[Cavanagh, 1999] P.Cavanagh, Top-Down Processing in Vision, *The MIT Press Encyclopedia of the Cognitive Sciences*, pp. 844–845.

[Jain, *et al.*, 1995] R.Jain, R.Kasturi, and B.G.Schunk, *Machine Vision*, McGraw-Hill, 1995.

[Josephson & Josephson, 1994] J.R.Jospehson and S.G.Josephson, *Abductive Inference: Computation, Philosophy, Technology*, Cambridge University Press, 1994.

[Levesque, 1996] H.Levesque, What Is Planning in the Presence of Sensing? *Proceedings AAAI 96*, pp. 1139–1146.

[Poole, 1993] D.Poole, Logic Programming, Abduction and Probability: A Top-Down Anytime Algorithm for Estimating Prior and Posterior Probabilities, *New Generation Computing*, vol. 11 (1993), pp. 377–400.

[Poole, 1995] D.Poole, Logic Programming for Robot Control, *Proceedings IJCAI 95*, pp. 150–157.

[Randell, *et al.*, 2001] D.Randell, M.Witkowski and M.Shanahan, From Images to Bodies: Modeling and Exploiting Spatial Occlusion and Motion Parallax, *Proceedings IJCAI 2001*, pp. 57–63.

[Scherl & Levesque, 1993] R.Scherl and H.Levesque, The Frame Problem and Knowledge Producing Actions, *Proceedings AAAI 93*, pp. 689–695.

[Shanahan, 1996] M.P.Shanahan, Robotics and the Common Sense Informatic Situation, *Proceedings ECAI 96*, pp. 684–688.

[Shanahan, 1997] M.P.Shanahan, Noise, Non-Determinism and Spatial Uncertainty, *Proceedings AAAI 97*, pp. 153–158.

[Shanahan, 2000] M.P.Shanahan, Reinventing Shakey, in *Logic-Based Artificial Intelligence*, ed. J.Minker, Kluwer Academic Press (2000), pp. 233–253.

Abducing Qualitative Spatio-Temporal Histories from Partial Observations

Shyamanta M Hazarika and **Anthony G Cohn**
School of Computing
University of Leeds
Leeds LS2 9JT, United Kingdom
e-mail{smh,agc}@comp.leeds.ac.uk

Abstract

We present a logic-based framework in which a qualitative spatio-temporal world model is constructed from *local surveys* i.e. partial spatio-temporal knowledge. Complete space-time *histories* are obtained through an abductive process driven by continuity constraints and a library of possibly domain specific behaviour patterns. The abduction technique is circumscription, which implements the heuristic that changes should only occur when forced to. We then discuss various additional heuristics to drive the selection of preferred explanations.

1 Introduction

In the field of robotic navigation, high precision metric approaches [23, 19] demand very reliable and accurate sensor devices. In contrast, over the years, *qualitative approaches* to robotic navigation have been proposed to overcome problems regarding complexity and stability. Qualitative modelling for self-localization and navigation includes the qualitative map approach [18, 20] and adaptive topological models [27]. They are based on a semantic hierarchy of descriptions for representation, with the most abstract level being that of the 'topological neighbourhood'. These approaches concern spatial learning and path planning in the absence of a single global coordinate system. However they do not exploit the notion of qualitative motion[1] per se for navigation.

Another important issue in robotic navigation is that of *anchoring*, the process of assigning abstract symbols to real sensor data [8]. The anchoring process establishes relation between the symbolic representation and the real world. This bridges the physical world and the states of a mobile agent. Logic based formalisms [34, 35, 36] have been used for tasks such as sensor data assimilation, map building and planning.

Following [34], we take a knowledge representation view of the connection between sensor information and states of a robot. The main aim of this paper is to present an approach to the problem of acquiring a qualitative world description from partial qualitative spatio-temporal (*henceforth s-t*) information such as might be acquired by a mobile agent exploring some region in space[2]. This is achieved by exploiting qualitative motion and a library of possible s-t patterns within a s-t theory and using circumscription to abduce explanations of the sensor data.

Exploiting Qualitative Motion

Within Qualitative Spatial Reasoning (QSR) (see [7] for an overview), navigation in true qualitative terms has been attempted by [32, 10]. Orientation knowledge can be seen as dynamic path knowledge and a qualitative concept of motion has been introduced [12]. Using Constraint Logic Programming over finite domains, a Qualitative Navigation Simulator for successful autonomous navigation of a simulated robot through an unknown structured environment has been built [10]. More recently, Bennett et al. [2] have explored the expressive power of a qualitative region-based geometry [3] and have applied it to the problem of representing and reasoning about the motion of rigid bodies within a confining environment.

[1] Motion can be seen as a form of spatio-temporal change and is interpreted thus in this paper.

[2] This can be seen as an abstract level of anchoring above the levels of anchoring put forward in [36]. We assume (in this analysis) the ability to anchor specific regions in the visual image field of the robot to named objects.

Using a Spatio-Temporal Ontology

With space-time as the ontological primitive, we exploit the notion of qualitative change and s-t continuity for building a world model as a step towards qualitative navigation. Our underlying spatial entities are extended regions of space-time [14, 26]. Following [14], space-time regions traced by objects over time are termed space-time *histories*. Fig. 1 shows a space-time history for a 2-D object and a *temporal slice* of the history. For n-D space, a space-time history is a n+1 dimensional volume (where $1 \leq n \leq 3$).

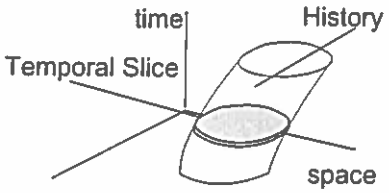


Figure 1: A s-t history is a n+1 dimensional volume for n-D space.

We present a logic-based framework in which a qualitative world model is constructed based on s-t histories obtained through an abductive process within an integrated s-t framework using s-t patterns (as in fig. 4) known a priori. Even though we emphasize here the use of an abductive framework for qualitative navigation in robotics and focus on an explicit notion of qualitative accounts of motion, such a framework could be used for other purposes: e.g., abducing complete histories from a sequence of temporally separated snapshots of a region or scene.

Shanahan was the first to propose map-building for robotic navigation as a formal abduction task [34]. He proposed a logic-based framework (based on circumscriptive event calculus) using abduction for sensor data assimilation. It falls short of using a purely qualitative approach to spatial representation [28] and space is represented as real-valued coordinate system.

As with any abductive process, a problem of Shanahan's approach is the existence of many minimal models that explain the agent's experiences [34]. This problem has been alleviated using the Spatial Semantic Hierarchy [18] as the agent's target spatial representation [29]. In line with Shanahan's suggestion [34, pages 34–35], grounded at the sensory level, we explore the use of a s-t version of the qualitative spatial representation language *Region Connection Calculus* (RCC) [28] for building a world model from sensory information.

Organization of the Paper

The paper is organized as follows. In section 2 we introduce the adbuctive framework. We discuss the spatio-temporal theory (based on the spatial representation language RCC-8) which forms the background theory for abduction and the spatio-temporal patterns – a set of typical patterns of behaviour of objects in the domain. The abductive reasoning engine is driven by selection heuristics. The selection heuristics are discussed in section 3. In section 4 we conclude and briefly discuss future directions.

2 Abductive Framework

A map emphasizes the illusion of seeing a spatial scene from *above* in an instant of time (a *snapshot*) which we refer to as *global snapshot*: the complete knowledge of the world at a time instant. In contrast, the knowledge of the world an autonomous agent garners as it continuously *explores* is only partial and we refer to it as *local survey* knowledge: partial spatial knowledge of the world at all times. With s-t primitives in an *inhabited* dynamic system[3], the key idea is to generate complete space-time *histories* by abduction: given a record of *local surveys*, the abductive task is to hypothesize the space-time *histories*, which, given the *s-t patterns* of objects in the domain and the background s-t theory, would explain the *local surveys*.

In logical terms, if a *local survey* is represented as the conjunction Φ of a set of s-t relationships, the task is to find an explanation of Φ in the form of a logical description (a mereotopological world model) Δ_H involving space-time *histories*, such that

$$\Sigma_{ST} \wedge \Delta_P \wedge \Delta_H \models \Phi \text{ , where}$$

1. Σ_{ST} is a s-t theory for space, time, change and continuity.
2. Δ_P is a logical description of *s-t patterns* for objects in the domain.

Fig. 2 below illustrates the abductive framework. The abductive reasoning engine is driven by selection heuristics. The process is a multi-tier procedure wherein explanations are abduced and then heuristics are used to choose preferred explanations. In this paper we first present a primary heuristic for spatial ab-

[3]Our interpretation of a dynamical system is as in [31]. A dynamical system is one whose state changes over time and where effects flow forwards in time. It is *inhabited* iff it contains one or more *agents* which can influence the system's state at later times by performing *actions*.

duction and then discuss a range of possibilities for refining the set of abduced explanations.

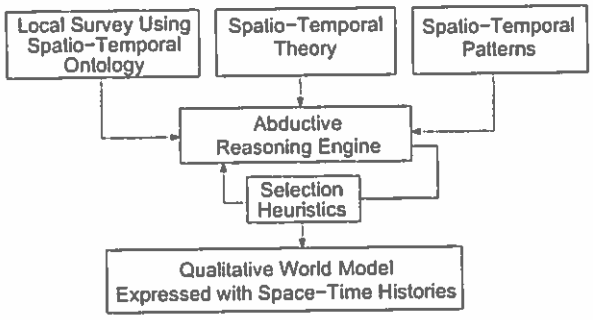


Figure 2: Abductive approach to generating an integrated s-t representation.

2.1 Spatio-Temporal Theory Σ_{ST}

We restrict discussion of the s-t theory Σ_{ST} here to s-t histories and continuity expressed in a purely mereotopological framework. Elsewhere we have discussed the theory in detail [6, 15]. We use three connection relations: C_{st}, C_{sp} and C_t, for *s-t*, *spatial* and *temporal* connection respectively. The axiomatisation of these connection relations are identical and follows [4]. From the topological primitive of $C_\alpha xy$ (where $\alpha \in \{st, sp, t\}$) we define the mereological relation of parthood, $P_\alpha xy$: x is a part of y.

The parthood relation P_α is used to define proper-part (PP_α), overlap (O_α) and discrete (DR_α). $DC_\alpha, EC_\alpha, PO_\alpha, EQ_\alpha, TPP_\alpha$ and $NTPP_\alpha$ i.e., disconnected, externally connected, partial overlap, equal, tangential proper part and non-tangential proper part respectively can also be defined. These relations along with the inverses for the last two viz. $TPPi_\alpha$ and $NTPPi_\alpha$ constitute the eight JEPD relations of the qualitative spatial representation language RCC-8 (see [4] for definitions). These relations are illustrated in fig 3 for the (2D) spatial case.

In order to introduce a s-t interpretation we must capture a notion of temporal order between the entities. For *temporal order* we write $x <_t y$: the closure of x strictly precedes the closure of y in time. From temporal connection and temporal order we can give the definition of *meets* ($\bowtie_t$) and all the temporal relations of Allen [1]. This includes relations such as *starts with* ($|\sqsubset_t$), *ends with* ($\sqsupset|_t$), *temporal inclusion* ($\subseteq_t$) and *temporal equivalence* ($\equiv_t$). Further, we define a three place relation of *temporal betweenness* ($z_2 \parallel_t (z_1; z_3)$ meaning z_2 is met by z_1 and meets z_3). We introduce relationships to refer to the initial and final parts of a history. $IPxy$ states that x is an initial part of history y just in case x starts with y and ends before it. Conversely, x is a final part of history y ($FPxy$) just in case x starts after y and ends with it. To define relations between space-time regions that may vary through time, we introduce the notion of a *temporal slice*, $TSxy$, i.e., x is a maximal component part of y corresponding to a certain time extent. We use the syntactic sugar $\frac{y}{w}$ to denote the part of y corresponding to the lifetime of w when it exists (i.e. when $w \subseteq_t y$). We also talk of slices through a collection of s-t histories (see [6] for details).

In expressions such as $\frac{x}{z}$ we will often talk of z as an *interval*, even though it is a s-t history, to emphasize that we are only interested in its temporal properties in this context. We do not in fact need purely temporal intervals in our ontology.

In [15] we defined space-time continuity. We say that a history x is *strongly* s-t continuous if it is 'spatially' and 'temporally' continuous simultaneously; we denote this as $StrCONTx$. Intuitively, this notion is the same as Muller's [26]: any space-time region is defined as qualitatively continuous just in case it is temporally self-connected and it doesn't make any spatial leaps. However this definition of continuity is unable to stop histories from 'temporal pinching', i.e. to exclude histories that appear and reappear again instantaneously at the same spatial location. In order to enforce a stronger notion of s-t continuity for histories, we disallow temporal pinching and introduce the notion of *firm* continuity. A *non-pinched* continuous s-t history is *firmly* continuous ($FCONTx$). The strongest notion of space-time continuity will be both FCONT and StrCONT; we denote this as $StrFCONTx$. Within our s-t theory we can define different notions of continuity based on how components connect over space-time (see complete analysis in [15]).

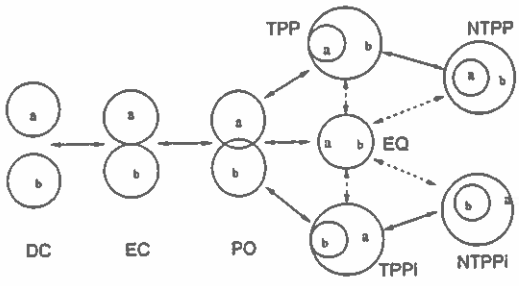


Figure 3: Transition Graph for RCC-8 relations

Under the strongest notion of space-time continuity (StrFCONT) fig. 3 shows the direct continuous transitions possible for the RCC-8 spatial relations. Any transition between RCC-8 relations to be consistent

with the background theory Σ_{ST} must follow a path through this graph. This imposes constraints on the abduction as discussed in section 2.3. If we assume that all objects are rigid, we can add an axiom which effectively constrains the possible continuous transitions in fig 3.

A1. $\mathsf{PP}\frac{x}{z_1}\frac{y}{z_1} \rightarrow \neg\mathsf{EQ}\frac{x}{z_2}\frac{y}{z_2}$

The illegal transitions are shown with dashed lines in fig. 3. This axiom will also disallow the sequence TPP $-\triangleright$ PO $-\triangleright$ TPPi which would otherwise be possible given the purely local constraints imposed by the restricted form of fig 3.

2.2 Spatio-Temporal Patterns Δ_P

The range of phenomena that can be described in a s-t theory of space is infinite. Identifying useful s-t patterns Δ_P involving one or more spatial entities is a complex task and one far beyond the scope of the present paper, where we are simply concerned with validating our abductive framework. Therefore here we enumerate only a small possible representative group of s-t patterns for rigid (shape invariant) objects which will be sufficient to illustrate the ideas. The following qualitative s-t patterns are identifiable with a single spatial entity:

1. Immobility $\mathsf{IMB}x$: Immobility is the phenomenon of occupying the *same space* at *all times*.
2. Non-cyclicity $\mathsf{NYC}x$: Non-cyclicity is defined as the phenomenon of *never* being in the *same place twice*.
3. Cyclicity $\mathsf{CYC}x$: Cyclicity is the phenomenon of being in the *same place more than once* (but *not* being IMB)[4].

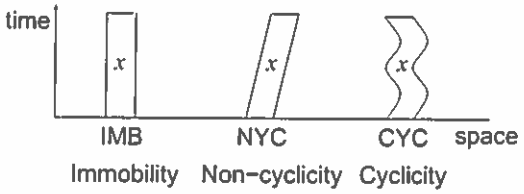


Figure 4: A selection of spatio-temporal patterns for a single entity.

Fig. 4 shows the different s-t patterns identified above. D1 to D3 provide the object level definitions for the above three different patterns. We cannot distinguish rotation from cyclic mobility. Topologically, they both have the property that objects are in the same place twice. However, with an additional morphological primitive of *congruence* [3] we could make the distinction[5].

D1. $\mathsf{IMB}x \equiv_{def} \forall t[t \subseteq_t x \rightarrow \mathsf{EQ}_{\mathsf{sp}}x\frac{x}{t}]$

D2. $\mathsf{NYC}x \equiv_{def} \forall u,v[(u \subseteq_t x \wedge v \subseteq_t x \wedge \neg(u \equiv_t v)) \rightarrow \neg\mathsf{EQ}_{\mathsf{sp}}\frac{x}{u}\frac{x}{v}]$[6]

D3. $\mathsf{CYC}x \equiv_{def} \exists u,v,w[(u \subseteq_t x \wedge v \subseteq_t x \wedge w \subseteq_t x \wedge v \parallel_t (u;w)) \wedge \mathsf{EQ}_{\mathsf{sp}}\frac{x}{u}\frac{x}{w} \wedge \neg\mathsf{EQ}_{\mathsf{sp}}\frac{x}{u}\frac{x}{v}]$

2.3 Transitions

In [6] three transition operators $\mathsf{TransTo}(r_1,r_3,x,y,z_1,z_2)$, $\mathsf{TransFrom}(r_1,r_2,x,y,z_1,z_2)$ and $\mathsf{InsRel3}(r_1,r_2,r_3,x,y,z_1,z_2)$ were defined. The first two operators assume that r_1 and r_3 hold over initial parts of z_1 and final parts of z_2 respectively and differ as to which of the two relations hold at the dividing instant. The third is for histories undergoing a transition between r_1 and r_3 with an instantaneous relation r_2 holding in between[7]. These definitions of $\mathsf{TransTo}(r_1,r_3,x,y,z_1,z_2)$ and $\mathsf{TransFrom}(r_1,r_2,x,y,z_1,z_2)$ did not always exactly characterise the required behaviour so we present slightly modified versions here:

D4. $\mathsf{TransTo}(r_1,r_3,x,y,z_1,z_2) \equiv_{def}$
$\forall u,v,p,q,m,n[(\mathsf{IP}u\frac{x}{z_1} \wedge \mathsf{TS}pu \wedge \mathsf{IP}v\frac{y}{z_1} \wedge \mathsf{TS}qv \wedge \mathsf{TS}m\frac{x}{z_2} \wedge \mathsf{TS}n\frac{y}{z_2}) \rightarrow (\mathsf{rcc}_{\mathsf{sp}}(r_1,p,q) \wedge z_1 \bowtie_t z_2 \wedge \mathsf{rcc}_{\mathsf{sp}}(r_2,m,n) \wedge \neg(r_1 = r_2))]$

D5. $\mathsf{TransFrom}(r_1,r_3,x,y,z_1,z_2) \equiv_{def}$
$\forall u,v,p,q,m,n[(\mathsf{FP}u\frac{x}{z_2} \wedge \mathsf{TS}pu \wedge \mathsf{FP}v\frac{y}{z_2} \wedge$

[4]There are weaker and stronger versions of these predicates possible too: eg. of never taking an overlapping path or of taking an overlapping path more than once (which might then yield the kind of semantic region descriptions computed in [11]).

[5]Further note that a sphere rotating occupies the same place at all times, so immobility as defined above does not necessarily mean being at rest. Also note that with congruence, we could make explicit at the object level the assumption that objects are rigid. Even without this we are able to axiomatize some of the effects of rigidity (see A1).

[6]Note that this definition implies that the object is never stationary. Objects that are in non-cyclic motion and at rest intermittently would display a combination of IMB and CYC over time. Thus could be expressed as a (macro) pattern explicitly if desired.

[7]For defining transitions (see [6] for details) between RCC relations, we treat RCC relations as constant symbols rather than as predicates; thus we define a predicate $\mathsf{rcc}(\psi,x,y)$: meaning Ψ holds between spatio-temporal regions x and y (where ψ is the lowercase translation of the RCC relation Ψ) i.e., $\mathsf{rcc}(\psi,x,y) \equiv \Psi(x,y)$. The transition operators are defined with the final two arguments to the rcc predicate as always co-temporal, so these amount to just testing the spatial topology $\mathsf{rcc}_{\mathsf{sp}}$.

$$\mathsf{TS}qv \wedge \mathsf{TS}m\tfrac{x}{z_1} \wedge \mathsf{TS}n\tfrac{y}{z_1}) \rightarrow (\mathsf{rcc_{sp}}(r_1, m, n) \wedge z_1 \bowtie_t z_2 \wedge \mathsf{rcc_{sp}}(r_2, p, q) \wedge \neg(r_1 = r_2))]$$

An *elementary transition* EleTran from an interval z_1 to an adjacent interval z_2 is defined as being a TransTo, TransFrom or an InsRel3 [6]. Thus any change occurring (excluding change in spatio-temporal pattern involving a single entity) involves an EleTran. For defining transitions between s-t patterns we treat pattern relations as constant symbols rather than as predicates. Thus we define a predicate $\mathsf{pat}(\mathsf{p}, x)$: meaning for x pattern $\mathfrak{P}$ holds; where p is the lowercase translation of the pattern relation $\mathfrak{P}$. We introduce the following definition schema:

D6. $\mathsf{pat}(\mathsf{p}, x) \equiv_{def} \mathfrak{P}(x)$

We now define *pattern transitions* for monadic s-t patterns. A pattern transition for a monadic pattern relation is specified by $\mathsf{PatTran}(x, p_1, p_2, z_1, z_2)$ where history x undergoes a transition of pattern from p_1 to p_2.

D7. $\mathsf{PatTran}(x, p_1, p_2, z_1, z_2) \equiv_{def} [\mathsf{pat}(p_1, \frac{x}{z_1}) \wedge \mathsf{pat}(p_2, \frac{x}{z_2}) \wedge z_1 \bowtie_t z_2 \wedge \neg(p_1 = p_2)]$

We define a *generalized transition* from an interval z_1 to an adjacent interval z_2 as being an elementary transition or a pattern transition.

D8. $\mathsf{GTran}(z_1, z_2) \equiv_{def} \exists x, y, r_1, r_2, r, p_1, p_2[\mathsf{PatTran}(x, p_1, p_2, z_1, z_2) \vee \mathsf{EleTran}(r_1, r, r_2, x, y, z_1, z_2)]$

It is important to capture the relationship between mobility and change. We now present two axioms (A2 and A3) which capture such properties. If an object occupies distinct regions of space at different times then it must be ¬IMB somewhere in between. We add axiom A2 to capture this constraint.

A2. $\neg\mathsf{EQ}\tfrac{x}{z_1}\tfrac{x}{z_2} \rightarrow \exists z_3[z_1 \leq_t z_3 \leq_t z_2 \wedge \neg\mathsf{IMB}\tfrac{x}{z_3}]$

Galton [13] has analysed what he terms 'states of motion' and the 'states of position'. He also introduces the concept of 'dominance': to say that q dominates p is to say that it is possible for q to hold at an instant which limits (at one or the other end) an open interval over which p holds. We introduce the predicate $\mathsf{Dom}(q, p)$ to express this. The 'states of motion' are dominated by the 'states of position'. In the case of $\mathsf{RCC_{sp}}$ relations we have that Dom(ec,dc), Dom(ec,po), Dom(tpp,po), Dom(tppi,po), Dom(tpp,ntpp), Dom(tppi,ntppi), Dom(eq,po), Dom(eq,tpp), Dom(eq,tppi), Dom(eq,ntpp), Dom(eq,ntppi). This notion imposes further restrictions on the TransTo and TransFrom predicates (InsRel3 already explicitly takes account of them). We add the following axiom to take account of this.

A3. $[\mathsf{rcc_{sp}}(r_1, \frac{x}{z_1}, \frac{y}{z_1}) \wedge \mathsf{rcc_{sp}}(r_2, \frac{x}{z_2}, \frac{y}{z_2}) \wedge \mathsf{Dom}(r_1, r_2) \wedge (z_1 \bowtie_t z_2 \vee z_2 \bowtie_t z_1)] \rightarrow [\neg\mathsf{IMB}\tfrac{x}{z_2} \vee \neg\mathsf{IMB}\tfrac{y}{z_2}]$

2.4 Specifying the Local Survey, Φ

Now that we have described our s-t theory Σ_{ST} and our pattern language Δ_P, it is appropriate to further specify the format of the observation Φ that we are assuming. We will confine ourselves to considering the observations of a single agent, which records, at a sequence of intervals $\tau_1, \tau_2, \ldots \tau_n$ the $\mathsf{rcc_{sp}}$ relationships between pairs of objects that it observes and any patterns it notices. Thus Φ consists of a conjunction of atoms each of the form $\mathsf{rcc_{sp}}(r, \frac{x}{\tau_j}, \frac{y}{\tau_i})$ or $\mathsf{pat}(\mathsf{p}_1, \frac{x}{\tau_j})$.

We conjoin to Φ also the n-1 facts $\tau_j \bowtie_t \tau_{j+1}$ where $1 \leq j \leq n-1$. This has the effect of uniqueness of names assumption for the named interval constants. We may also add statements asserting the agent's belief of the continuity of the histories e.g. that a history x is strongly continuous i.e. $\mathsf{StrFCONT}(x)$. We also need to add axioms to express the uniqueness of names for the named objects in the observations. It is also helpful to assume that if the agent observes an object at a particular time interval, then it is able to observe everything about it (i.e. its pattern of behaviour during z_1 and the spatial relationships between it and all the objects it observes during z_1). We introduce a predicate $\mathsf{Obv}(t)$ to mean t is an observation interval. We will also add the assumption that all objects have the same lifetime. Thus Φ consists of:

1. A conjunction $\mathsf{Obv}(\tau_1) \wedge \ldots \wedge \mathsf{Obv}(\tau_n) \wedge \tau_1 \bowtie_t \tau_2 \wedge \ldots \wedge \tau_{n_1} \bowtie_t \tau_n$ where $\tau_1, \ldots, \tau_n$ are constant symbols denoting the observation intervals of the agent. Note that unless $n \geq 2$ then no change can occur; so we assume $n \geq 2$.
2. A conjunction of atomic facts Ω expressing observed s-t knowledge at the τ_i, e.g. $\mathsf{DC_{sp}}\tfrac{b}{\tau_2}\tfrac{c}{\tau_2}, \mathsf{EQ_{sp}}\tfrac{a}{\tau_1}\tfrac{a}{\tau_4}, \mathsf{IMB}\tfrac{a}{\tau_2}$
3. A conjunction $\bigwedge^{x_i \in \Gamma} \mathsf{StrFCONT}(x_i)$ stating that each of the objects in Γ is strongly continuous, where Γ is the set of named objects in Ω.
4. An axiom expressing the uniqueness of names of the objects in Γ.
5. The assumption that all objects have the same lifetime: $\bigwedge^{\alpha,\beta \in \Gamma} \alpha \equiv_t \beta$
6. We assume that during every observation interval each history follows one of the behaviour patterns.

This can be expressed by the following axiom.

A4. $\mathsf{Obv}(z) \rightarrow \exists p[\mathsf{pat}(p, \frac{x}{z})]$

3 Selection Heuristics

In general, abduction may yield more than one possible answer. Often abductive reasoning is accompanied by some preference criteria [30]. We can express these criteria using heuristics[8].

Our primary heuristic is to choose the explanations that minimize the number of changes of state i.e. exploit some form of 'global' persistence (or *s-t inertia*).

3.1 Global Persistence

Most non-monotonic approaches to reasoning about time and change assume that *fluents* tend to persist if nothing tells us the contrary. In other words, unnecessary change is minimized. We would like to import this *inertia* assumption into our s-t theory. Prior to this we need to make a few more definitions and state certain assumptions.

We wish to characterize change in an abstract and qualitative way. We will only consider qualitative change between named histories. Thus purely metric changes which do not result in a qualitative change do not effect the explanations generated, nor do changes involving histories not corresponding to a named object of interest.

Within our s-t theory, a formulae f_{st} (involving spatial relations between one or more space-time histories) whose value evolves over different temporal slices is a *spatio-temporal fluent*. Our logical language for describing s-t fluents is $\Sigma_{ST} \cup \Delta_P$. Spatio-temporal inertia is expressed by the following reasoning step:

> Given that a s-t fluent f_{st} holds during a given slice z_1, can we conclude that it holds during the subsequent slice z_2 (where $z_1 \bowtie_t z_2$)? E.g. if $\mathsf{IMB}\frac{x}{z_1}$ then is $\mathsf{IMB}\frac{x}{z_2}$ true ?

There are two different categories of answers to this problem. Provided the s-t fluent f_{st} is known to be monotonic a priori, abductive inferences under *local survey* do not cause particular problem[9]. It is for abductive inference using non-monotonic s-t fluents that the inertia assumption needs to be exploited.

Definition 1 *Spatio-temporal change: Given a collection of named s-t histories, a s-t change occurs if z_1 and z_2 are named slices through these histories (with $z_1 \bowtie_t z_2$) and $\mathsf{GTran}(z_1, z_2)$ is true.*

Definition 2 *Episode: Given a collection of named s-t histories, an* episode *is the maximal slice through all these histories during which no s-t change occurs. We introduce the predicate $\mathsf{Episode}(e_i)$ to denote this notion* (D9).

D9. $$\mathsf{Episode}(e_i) \equiv_{def} \exists z_1, z_2[\mathsf{Obv}(z_1) \wedge \mathsf{Obv}(z_2) \wedge z_1 \,|\sqsubset_t e_i \wedge z_2 \sqsupset|_t e_i] \wedge \neg\exists z_1, z_2[\mathsf{Obv}(z_1) \wedge \mathsf{Obv}(z_2) \wedge z_1 \subseteq_t e_i \wedge z_2 \subseteq_t e_i \wedge z_1 \bowtie_t z_2 \wedge \mathsf{GTran}(z_1, z_2)]$$

Definition 3 *Episodic Boundary: Given two episodes e_i and e_j such that $e_i \bowtie_t e_j$, the the pair (e_i, e_j) is an* episodic boundary. *We introduce the predicate $\mathsf{EB}(e_i, e_j)$ to denote this notion.*

D10. $$\mathsf{EB}(e_1, e_2) \equiv_{def} \mathsf{Episode}(e_1) \wedge \mathsf{Episode}(e_2) \wedge e_i \bowtie_t e_j$$

Note that although an object may be moving during some interval this does not necessarily imply there is any s-t change in our framework. E.g. we can have $\mathsf{NYC}\frac{b}{z}$ but no episode boundaries need occur during z unless some RCC_{sp} relationship involving b and another object changes.

The Circumscriptive Theory $\mathfrak{CT}$

Circumscription is a form of nonmonotonic reasoning initially introduced by McCarthy [24] and further developed [21] for reasoning under incomplete information. The basic idea of circumscription is to limit the set of objects of which a predicate is true, a process which is known as *minimising* the predicate.

Let ρ_1 and ρ_2 be predicates with arity n. Let $\bar{x}$ be a tuple of n distinct variables. We have the following notation.

- $\rho_1 = \rho_2$ means $\forall \bar{x}[\rho_1(\bar{x}) \leftrightarrow \rho_2(\bar{x})]$
- $\rho_1 \leq \rho_2$ means $\forall \bar{x}[\rho_1(\bar{x}) \rightarrow \rho_2(\bar{x})]$
- $\rho_1 < \rho_2$ means $[\rho_1 \leq \rho_2] \wedge \neg[\rho_1 = \rho_2]$

Let $\mathsf{A}(\mathsf{Ab}, \mathsf{Z}_1, \ldots \mathsf{Z}_m)$ be a sentence containing a predicate constant Ab and object, function, and/or predicate constants $\mathsf{Z}_1, \ldots, \mathsf{Z}_m$ (and possibly other object,

[8] In order to avoid trivial explanations, a set of predicates is distinguished such that every acceptable explanation must contain only these predicates. Further, given a theory Σ_{ST} and a formula Φ to be explained, we add conditions $\Sigma_{ST} \not\models \Phi$ and $\Sigma_{ST} \not\models \neg\Phi$ guaranteeing that the set of all explanations is non-empty and non-trivial [33].

[9] Monotonicity is convenient because it means we can reason about what an agent believes on the basis of partial knowledge about its beliefs [17].

function and predicate constants). The *circumscription* of Ab in A with *varied* $Z_1, ..., Z_m$ is the sentence

$$A(Ab, Z_1, ..., Z_m) \wedge \neg\exists ab, z_1, ..., z_m [A(ab, z_1, ..., z_m) \wedge ab < Ab]$$

Here ab is a predicate variable of the same arity as Ab; if Z_i is an object constant, then z_i is an object variable and if Z_i is a function/predicate constant, then z_i is a function/predicate variable of the same arity. The equality symbol is not allowed to appear in the list $Z_1, ..., Z_m$. If Z denotes the tuple $Z_1, ..., Z_m$ and z denotes the tuple $z_1, ..., z_m$; then the above formula can be written as

$$A(Ab, Z) \wedge \neg\exists ab, z [A(ab, z) \wedge ab < Ab]$$

The subformula $\neg\exists ab, z[A(ab, z) \wedge ab < Ab]$ says that the extent of Ab is minimal. Minimality is understood as the impossibility of making the extent of the circumscribed predicate smaller even when some of the object, function, or predicate constants occurring in A are allowed to vary along with Ab in the process of minimizing its extent. The above formula is denoted as CIRC[A; Ab; Z].

Minimizing Spatio-Temporal Change

Some changes are forced by the observations, e.g. if $\{DC(\frac{x}{z_1}, \frac{y}{z_1}), EC(\frac{x}{z_2}, \frac{y}{z_2}), z_1 \bowtie_t z_2\} \subseteq \Phi$ then $GTran(z_1, z_2)$ is forced. However, if $\{IMB\frac{x}{z_1}, IMB\frac{x}{z_3}, (z_2 \parallel_t (z_1; z_3))\} \subseteq \Phi$ and nothing else is known about x in Φ then we want to assume that there is no change of pattern for x in z_2. This is akin to the commonsense law of inertia in our framework.

Spatio-temporal inertia is achieved by minimizing GTran and thus the number of episodes. This is done by posing the problem as a *circumscriptive theory* under minimization of generalized transition GTran[10].

A5. Σ_{ST}

A6. Δ_P

A7. Φ

P1. **circ** GTran **var** Λ

where Λ is the set of predicates that may occur in s-t fluents.

Example 1: Let us assume the scenarios as shown in fig. 5 for an autonomous agent, a, with on-board vision in an inhabited environment[11].

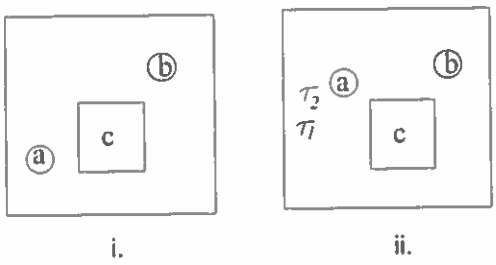


Figure 5: Scenarios of an *inhabited* dynamic environment: (i.) initial state, (ii.) final state showing path with dotted lines. Two observation intervals τ_1 and τ_2 are represented in (ii).

There are two qualitatively different temporal parts: initially (during τ_1) a sees only c, then (during τ_2) it sees b as well. Thus during τ_1 we have[12]

$$\{DC\frac{a}{\tau_1}\frac{c}{\tau_1}, IMB\frac{c}{\tau_1}, NYC\frac{a}{\tau_1}\} \subseteq \Phi$$

and during τ_2 we have

$$\{DC\frac{a}{\tau_2}\frac{c}{\tau_2}, DC\frac{a}{\tau_2}\frac{b}{\tau_2}, DC\frac{b}{\tau_2}\frac{c}{\tau_2}, IMB\frac{c}{\tau_2}, IMB\frac{b}{\tau_2}, NYC\frac{a}{\tau_2}\} \subseteq \Phi$$

It can also record the *pure* spatial relationships between τ_1 and τ_2, i.e.,

$$\{EQ\frac{c}{\tau_1}\frac{c}{\tau_2}, PO\frac{a}{\tau_1}\frac{a}{\tau_2}\} \subseteq \Phi$$

Since $\Sigma_{ST} \cup \Delta_P \cup \Phi$ does not imply any change of pattern or RCC_{sp} relationship between τ_1 and τ_2, minimizing GTran will in fact result in an empty extension for GTran. Thus there are no episodic boundaries and the only explanation possible is[13]:

$$\Delta_H = [IMB\frac{b}{\tau_1} \wedge DC\frac{b}{\tau_1}\frac{c}{\tau_1} \wedge DC\frac{a}{\tau_1}\frac{b}{\tau_1} \wedge \neg GTran(\tau_1, \tau_2)] \quad \text{(E1)}$$

The first conjunct is of particular interest in this example. $IMB\frac{b}{\tau_1}$ is abduced from the observation $IMB\frac{b}{\tau_2}$

[10] The circumscriptive theory here is defined using the notation in [21, pages 307-308]. Instead of CIRC[A; Ab; Z], the theory axioms A (A1 to A7) are listed, followed by the circumscription policy **circ** Ab **var** Z (P1).

[11] We are not concerned here with issues of lower level vision such as segmentation and recognition of objects. We assume that such lower level vision algorithms are available. As pointed out in section 1, we assume an ability to anchor specific regions in the robot's visual image field to named objects.

[12] In the presentation of the examples when we write an RCC relationship such as $EC_{sp}\frac{a}{\tau_1}\frac{b}{\tau_2}$ we will omit the sp subscript for notational convenience.

[13] Note that for those observations which do not change during Φ (e.g. the RCC_{sp} relation between a and c), no explanation is produced. A solution is to add an initial observation interval τ_0, without any observations. The abduction procedure would then abduce that the values must also hold in τ_0. Thus the τ_0 values would also appear in the abduced formulae Δ_H. The explanation of any observation which do not change during Φ, e.g. $DC_{sp}\frac{a}{\tau_1}\frac{c}{\tau_1}$ would thus be that a and c were DC_{sp} just before τ_1 and $\neg EB(\tau_0, \tau_1)$.

based on an empty extension for GTran. Of course if we did not make the assumption, in section 2.4, that all objects have the same lifetime, then other explanations might be possible (provided we extended the notion of GTran to incorporate changes owing to objects coming into existence and ceasing to exist). Also note that if **a** was involved in some change from τ_1 to τ_2 (e.g. $\mathsf{DC}\frac{\mathbf{a}}{\tau_1}\frac{\mathbf{c}}{\tau_1}$, $\mathsf{EC}\frac{\mathbf{a}}{\tau_2}\frac{\mathbf{c}}{\tau_2}$) then an episodic boundary would be forced, and changes involving **b** could occur 'for free', thus resulting in multiple explanations (e.g. where **b** starts moving during τ_2). Later, in section 3.2, we discuss how these might be avoided.

Example 2: Consider another scenario for the autonomous agent **a** as shown in fig. 6.

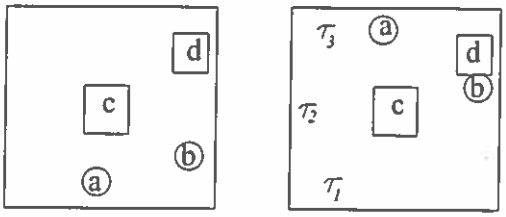


Figure 6: Another set of scenarios for autonomous agent **a** in an inhabited dynamic environment. (i.) initial state, (ii.) final state showing path with dotted lines. Three observation intervals τ_1, τ_2 and τ_3 are represented in (ii).

There are three qualitatively different observation intervals : initially (during τ_1) **a** sees **b** and **c**. Thus we have

$$\{\mathsf{DC}\tfrac{\mathbf{a}}{\tau_1}\tfrac{\mathbf{b}}{\tau_1},\ \mathsf{DC}\tfrac{\mathbf{a}}{\tau_1}\tfrac{\mathbf{c}}{\tau_1},\ \mathsf{DC}\tfrac{\mathbf{c}}{\tau_1}\tfrac{\mathbf{b}}{\tau_1},\ \mathsf{IMB}\tfrac{\mathbf{c}}{\tau_1},\ \mathsf{NYC}\tfrac{\mathbf{b}}{\tau_1},\ \mathsf{NYC}\tfrac{\mathbf{a}}{\tau_1}\} \subseteq \Phi$$

Then (during τ_2) it sees only **c** i.e,

$$\{\mathsf{DC}\tfrac{\mathbf{a}}{\tau_2}\tfrac{\mathbf{c}}{\tau_2},\ \mathsf{IMB}\tfrac{\mathbf{c}}{\tau_2},\ \mathsf{NYC}\tfrac{\mathbf{a}}{\tau_2}\} \subseteq \Phi$$

and finally (during τ_3) it sees **b** again and also **d** for the first time:

$$\{\mathsf{DC}\tfrac{\mathbf{a}}{\tau_3}\tfrac{\mathbf{b}}{\tau_3},\ \mathsf{DC}\tfrac{\mathbf{a}}{\tau_3}\tfrac{\mathbf{c}}{\tau_3},\mathsf{DC}\tfrac{\mathbf{a}}{\tau_3}\tfrac{\mathbf{d}}{\tau_3},\ \mathsf{DC}\tfrac{\mathbf{b}}{\tau_3}\tfrac{\mathbf{c}}{\tau_3},\ \mathsf{EC}\tfrac{\mathbf{b}}{\tau_3}\tfrac{\mathbf{d}}{\tau_3},\ \mathsf{DC}\tfrac{\mathbf{c}}{\tau_3}\tfrac{\mathbf{d}}{\tau_3},\ \mathsf{IMB}\tfrac{\mathbf{b}}{\tau_3},\ \mathsf{IMB}\tfrac{\mathbf{c}}{\tau_3},\mathsf{IMB}\tfrac{\mathbf{d}}{\tau_3},\ \mathsf{NYC}\tfrac{\mathbf{a}}{\tau_3}\} \subseteq \Phi$$

It can also record the *pure* spatial relationships. Thus we also have the following:

$$\{\mathsf{EQ}\tfrac{\mathbf{c}}{\tau_1}\tfrac{\mathbf{c}}{\tau_2},\ \mathsf{EQ}\tfrac{\mathbf{c}}{\tau_2}\tfrac{\mathbf{c}}{\tau_3},\ \mathsf{PO}\tfrac{\mathbf{a}}{\tau_1}\tfrac{\mathbf{a}}{\tau_2},\ \mathsf{PO}\tfrac{\mathbf{a}}{\tau_2}\tfrac{\mathbf{a}}{\tau_3},\ \mathsf{DC}\tfrac{\mathbf{b}}{\tau_1}\tfrac{\mathbf{b}}{\tau_3}\} \subseteq \Phi$$

With these observations, based on the s-t patterns and minimization of GTran, we have a single episodic boundary (i.e. 2 episodes) though it may occur either after τ_1 or after τ_2. The following formula is one possible explanation of the *local survey* made by **a**:

$$\Delta_H = [\mathsf{IMB}\tfrac{\mathbf{b}}{\tau_2} \wedge \mathsf{IMB}\tfrac{\mathbf{d}}{\tau_2} \wedge \mathsf{DC}\tfrac{\mathbf{a}}{\tau_2}\tfrac{\mathbf{b}}{\tau_2} \wedge \mathsf{DC}\tfrac{\mathbf{c}}{\tau_2}\tfrac{\mathbf{d}}{\tau_2} \wedge \mathsf{EC}\tfrac{\mathbf{b}}{\tau_2}\tfrac{\mathbf{d}}{\tau_2} \wedge \mathsf{GTran}(\tau_1, \tau_2)] \quad \textbf{(E2)}$$

Alternatively the episode boundary may occur after τ_2 rather than after τ_1:

$$\Delta_H = [\mathsf{NYC}\tfrac{\mathbf{b}}{\tau_2} \wedge \mathsf{DC}\tfrac{\mathbf{a}}{\tau_2}\tfrac{\mathbf{b}}{\tau_2} \wedge \mathsf{DC}\tfrac{\mathbf{c}}{\tau_2}\tfrac{\mathbf{d}}{\tau_2} \wedge \mathsf{GTran}(\tau_2, \tau_3)] \quad \textbf{(E3)}$$

Note that in neither of these explanations can we infer knowledge about **d** before the episode boundary. We will return to this issue in section 3.2.

3.2 Additional Heuristics

In the preceding section we have shown how circumscribing GTran addresses the issue of 'global' s-t inertia. As we have seen, in some very simple cases this may be sufficient to generate a unique explanation. However, in general, multiple explanations will still be possible. In this section we explore some further heuristics which might be used to prefer one explanation to another. Here are some such possible heuristics:

1. Prefer explanations where change happens as late as possible (i.e., the initial state extends for as long as possible)
2. Prefer explanations where change happens as early as possible (i.e., the final state extends as far back in time as possible)
3. Prefer explanations where the total 'number of changes' is minimal (i.e., if we count the number of changes at each episode boundary, and sum these, then this sum is minimal). There are variants of this e.g. where one minimizes the number of changes at the last or the first episodic boundary.
4. We assume some a priori knowledge on the kinds of change which might occur (e.g. certain patterns are more likely and/or certain objects more likely to be immobile).

To implement the first heuristic above is fairly straightforward. We can define a predicate $\mathsf{NIIE}(z)$ (Not_In_Initial_Episode) which is true when z is one of the observation intervals and occurs after the initial episode. Circumscribing NIIE so as to minimize it will effectively maximize the extent of the initial episode and thus delay the first s-t change till as late as possible. Defining NIIE is straightforward:

D11. $\mathsf{NIIE}(z) \equiv_{def} \mathsf{Obv}(z) \wedge \exists z_1, z_2[\mathsf{EB}(z_1, z_2) \wedge z_1 <_t z]$

If the second heuristic were preferred (these presumably being mutually exclusive), then one could dually define a predicate $\mathsf{NIFE}(z)$ (Not_In_Final_Episode)

which is true when z is one of the observation intervals and occurs before the final episode. Circumscribing NIFE so as to minimize it would maximize the extent of the final episode and thus force the final change to occur as early as possible.

The third heuristic above is a little trickier to implement though it could be used in addition to either of the first two, and either before or after them (obviously with possibly different effects). First we need to define a predicate which is true when some change occurs, and which effectively names each of them separately (unlike $\mathsf{GTran}(z_1, z_2)$ which does not distinguish how many changes occur at the episodic boundary (z_1, z_2)). We also have to consider how to count the changes. Clearly some changes imply others; e.g. $\mathsf{EleTran}(\mathsf{ec}, \mathsf{dc}, x, y, z_1, z_2) \wedge \mathsf{IMB}\frac{x}{z_1} \wedge \mathsf{IMB}\frac{y}{z_1}$ implies $\neg\mathsf{IMB}\frac{x}{z_2}$ or $\neg\mathsf{IMB}\frac{y}{z_2}$ (from Axioms A2 and A3). In this case we might not want to double count such changes. For simplicity, if there is at least one change involving two histories x and y at an episodic boundary, we will just "count one". We will also "count one" for each monadic pattern change. Thus we define:

D12. $\mathsf{CTran}(x, y) \equiv_{def} \exists z_1, z_2, r_1, r_2, r_3, p_1, p_2$
$[\mathsf{EleTran}(x, y, r_1, r_2, r_3, z_1, z_2)$
$\vee\ \mathsf{PatTran}(x, p_1, p_2, z_1, z_2)]$

To achieve the desired affect of minimizing the number of changes over all episodic boundaries, we need to minimize the *cardinality* of the extension of CTran, rather than the weaker operation of simply minimizing the extension of the predicate as standard circumscription achieves. This can be achieved via *model minimization* [16], also called *variable domain circumscription* [22]. Note that this heuristic should allow the propogation of fluents which are not forced to change, even across episodic boundaries (cf the remark made at the end of Example 2).

To implement the fourth heuristic above, we need to introduce different sorts for various kinds of objects (e.g. $\mathsf{Agent}(x)$, $\mathsf{Physob}(x)$). We can then introduce default rules associating particular patterns of behaviour with particular sorts of objects, e.g. that agents usually are not immobile, but that physical objects are. This kind of default behaviour can be achieved in the usual way in a circumscriptive framework by introducing abnormality predicates which are then circumscribed.

Example 3: Let us consider another scenario as illustrated in fig 7 below.

There are five qualitatively different observation intervals. Initially (during τ_1) a sees c, d and b. Thus we

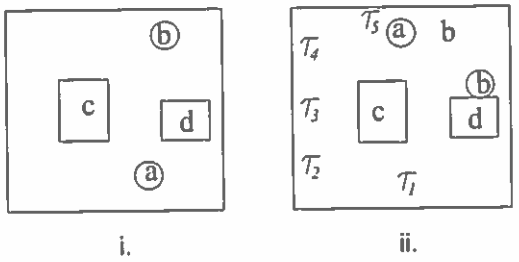


Figure 7: Set of scenarios for autonomous agent a in an inhabited dynamic environment. (i.) initial state, (ii.) final state showing path with dotted lines. Five observation intervals $\tau_1, \tau_2, \tau_3, \tau_4$ and τ_5 are represented in (ii).

have
$\{\mathsf{DC}\frac{a}{\tau_1}\frac{b}{\tau_1}, \mathsf{DC}\frac{a}{\tau_1}\frac{c}{\tau_1}, \mathsf{DC}\frac{a}{\tau_1}\frac{d}{\tau_1}, \mathsf{DC}\frac{b}{\tau_1}\frac{c}{\tau_1}, \mathsf{DC}\frac{b}{\tau_1}\frac{d}{\tau_1}, \mathsf{DC}\frac{c}{\tau_1}\frac{d}{\tau_1},$
$\mathsf{IMB}\frac{b}{\tau_1}, \mathsf{IMB}\frac{c}{\tau_1}, \mathsf{IMB}\frac{d}{\tau_1}, \mathsf{NYC}\frac{a}{\tau_1}\} \subseteq \Phi$

Then (during τ_2) it sees c and d but not b i.e,
$\{\mathsf{DC}\frac{a}{\tau_2}\frac{c}{\tau_2}, \mathsf{DC}\frac{a}{\tau_2}\frac{d}{\tau_2}, \mathsf{DC}\frac{c}{\tau_2}\frac{d}{\tau_2}, \mathsf{IMB}\frac{c}{\tau_2}, \mathsf{IMB}\frac{d}{\tau_2}, \mathsf{NYC}\frac{a}{\tau_2}\} \subseteq \Phi$

Thereafter during τ_3, it does not see anything other than c. Thus we have
$\{\mathsf{DC}\frac{a}{\tau_3}\frac{c}{\tau_3}, \mathsf{IMB}\frac{c}{\tau_3}, \mathsf{NYC}\frac{a}{\tau_3}\} \subseteq \Phi$

After that, during τ_4, it sees b as well as c:
$\{\mathsf{DC}\frac{a}{\tau_4}\frac{b}{\tau_4}, \mathsf{DC}\frac{a}{\tau_4}\frac{c}{\tau_4}, \mathsf{DC}\frac{b}{\tau_4}\frac{c}{\tau_4}, \mathsf{IMB}\frac{c}{\tau_4}, \mathsf{NYC}\frac{b}{\tau_4}, \mathsf{NYC}\frac{a}{\tau_4}\} \subseteq \Phi$

Finally during τ_5, it sees d again
$\{\mathsf{DC}\frac{a}{\tau_5}\frac{b}{\tau_5}, \mathsf{DC}\frac{a}{\tau_5}\frac{c}{\tau_5}, \mathsf{DC}\frac{a}{\tau_5}\frac{d}{\tau_5}, \mathsf{DC}\frac{b}{\tau_5}\frac{c}{\tau_5}, \mathsf{EC}\frac{b}{\tau_5}\frac{d}{\tau_5},$
$\mathsf{DC}\frac{c}{\tau_5}\frac{d}{\tau_5}, \mathsf{IMB}\frac{c}{\tau_5}, \mathsf{IMB}\frac{d}{\tau_5}, \mathsf{NYC}\frac{b}{\tau_5}, \mathsf{NYC}\frac{a}{\tau_5}\} \subseteq \Phi$

It can also record the *pure* spatial relationships. Thus we also have the following
$\{\mathsf{EQ}\frac{c}{\tau_1}\frac{c}{\tau_2}, \mathsf{EQ}\frac{c}{\tau_2}\frac{c}{\tau_3}, \mathsf{EQ}\frac{c}{\tau_3}\frac{c}{\tau_4}, \mathsf{EQ}\frac{c}{\tau_4}\frac{c}{\tau_5},$
$\mathsf{EQ}\frac{d}{\tau_1}\frac{d}{\tau_2}, \mathsf{PO}\frac{a}{\tau_1}\frac{a}{\tau_2}, \mathsf{PO}\frac{a}{\tau_2}\frac{a}{\tau_3}, \mathsf{PO}\frac{a}{\tau_3}\frac{a}{\tau_4}, \mathsf{PO}\frac{a}{\tau_4}\frac{a}{\tau_5},$
$\mathsf{PO}\frac{b}{\tau_3}\frac{b}{\tau_4}, \mathsf{PO}\frac{b}{\tau_4}\frac{b}{\tau_5}, \mathsf{DC}\frac{b}{\tau_1}\frac{b}{\tau_4}, \mathsf{DC}\frac{b}{\tau_1}\frac{b}{\tau_5}\} \subseteq \Phi$

Although there are four potential locations for episodic boundaries, circumscribing GTran results in only two episodes, with alternative locations for the episodic boundary as shown in the two explanations below:

$$\Delta_H = [\mathsf{NYC}\frac{b}{\tau_2} \wedge \mathsf{NYC}\frac{b}{\tau_3} \wedge \mathsf{IMB}\frac{d}{\tau_3} \wedge \mathsf{IMB}\frac{d}{\tau_4} \wedge \mathsf{DC}\frac{a}{\tau_2}\frac{b}{\tau_2} \wedge \mathsf{DC}\frac{a}{\tau_3}\frac{b}{\tau_3} \wedge \mathsf{DC}\frac{a}{\tau_3}\frac{d}{\tau_3} \wedge \mathsf{DC}\frac{a}{\tau_4}\frac{d}{\tau_4} \wedge \mathsf{DC}\frac{c}{\tau_3}\frac{d}{\tau_3} \wedge \mathsf{DC}\frac{c}{\tau_4}\frac{d}{\tau_4} \wedge \mathsf{GTran}(\tau_1, \tau_2)] \quad \textbf{(E4)}$$

$$\Delta_H = [\mathsf{IMB}\frac{b}{\tau_2} \wedge \mathsf{IMB}\frac{b}{\tau_3} \wedge \mathsf{IMB}\frac{d}{\tau_3} \wedge \mathsf{IMB}\frac{d}{\tau_4} \wedge \mathsf{DC}\frac{a}{\tau_2}\frac{b}{\tau_2} \wedge \mathsf{DC}\frac{a}{\tau_3}\frac{b}{\tau_3} \wedge \mathsf{DC}\frac{a}{\tau_3}\frac{d}{\tau_3} \wedge \mathsf{DC}\frac{a}{\tau_4}\frac{d}{\tau_4} \wedge \mathsf{DC}\frac{c}{\tau_3}\frac{d}{\tau_3} \wedge \mathsf{DC}\frac{c}{\tau_4}\frac{d}{\tau_4} \wedge \mathsf{GTran}(\tau_3, \tau_4)] \quad \textbf{(E5)}$$

3.2.1 Binary Behaviour Patterns

In our earlier presentation of spatial behaviour patterns we only considered monadic patterns involving a single s-t history. However, in general one might consider patterns involving two or more histories and if such behaviour patterns can be preferentially associated with particular sorts of objects, then this will provide additional heuristic knowledge to constrain possible explanations. In the case of pairs of spatial entities, x, y, fig. 8 shows some possible patterns for rigid objects which do not interpenetrate each other. These are:

1. Coalescence $\mathsf{COL}xy$: a *coming together* of two bodies for a period.
2. Separation $\mathsf{SEP}xy$: Separation of two bodies that have previously behaved as a unit for a period. This is the dual of coalescence.
3. Collision $\mathsf{CLN}xy$: a dynamic event when two bodies come into *contact and separate again.* A collision could be *instantaneous* or a *coalescence* followed by a *separation.*
4. Disjointness $\mathsf{DIS}xy$: Two bodies remain disjoint for a period.
5. Attachment $\mathsf{ATT}xy$: Two bodies remain attached for a period.

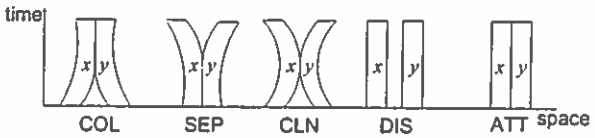


Figure 8: A selection of binary spatio-temporal patterns.

It is straightforward to define these patterns in terms of the existing apparatus except that in order to define $\mathsf{DIS}xy$, we define a predicate $\mathsf{IntP}xy$: x is an interior part of y.

D13. $\mathsf{IntP}xy \equiv_{def} \exists z_1, z_2[\mathsf{TS}xy \wedge \mathsf{IP}z_1y \wedge \mathsf{FP}z_2y \wedge x \parallel_t (z_1; z_2)]$

D14 through D16 provide the object level definitions for the first three binary spatio-temporal patterns. D17 and D18 define disjointness and attachment respectively.

D14. $\mathsf{COL}xy \equiv_{def} \exists u, v[u \mid\sqsubset_t x \wedge u \bowtie_t v \wedge v \sqsupset|_t x \wedge x \equiv_t y \wedge \mathsf{TransTo}(\mathsf{dc}, \mathsf{ec}, x, y, u, v)]$

D15. $\mathsf{SEP}xy \equiv_{def} \exists u, v[u \mid\sqsubset_t x \wedge u \bowtie_t v \wedge v \sqsupset|_t x \wedge x \equiv_t y \wedge \mathsf{TransFrom}(\mathsf{ec}, \mathsf{dc}, x, y, u, v)]$

D16. $\mathsf{CLN}xy \equiv_{def} \exists u, v, w[u \mid\sqsubset_t x \wedge v \parallel_t (u; w) \wedge w \sqsupset|_t x \wedge x \equiv_t y \wedge [[\mathsf{TransTo}(\mathsf{dc}, \mathsf{ec}, x, y, u, v) \wedge \mathsf{TransFrom}(\mathsf{ec}, \mathsf{dc}, x, y, v, w)] \vee \mathsf{InsRel3}(\mathsf{dc}, \mathsf{ec}, \mathsf{dc}, x, y, u, v)]]$

D17. $\mathsf{DIS}xy \equiv_{def} \forall u, v[[\mathsf{IntP}ux \wedge \mathsf{IntP}vy] \rightarrow \mathsf{DC}uv]$

D18. $\mathsf{ATT}xy \equiv_{def} \forall t[(t \subseteq_t x \wedge t \subseteq_t y) \rightarrow \mathsf{EC}_{\mathsf{sp}}\frac{x}{t}\frac{y}{t}]$

Clearly many other possible binary patterns are possible. E.g. Muller [25] presents other examples of binary patterns (e.g. crossing, leaving, entering) including cases where interpenetration occurs. One can think of many domain examples where such patterns might be prototypically associated with particular kinds of object pairs; for example in a woodworking domain, a nail and piece of wood would typically either have a COL or a DIS behaviour pattern. Similarly, Egenhofer [9] discusses how various patterns of behaviours for deformable objects can be associated with paths through a transition network (e.g. expanding, contracting). These could form possible (complex) patterns.

Our notion of generalised transition, GTran, was defined in terms of monadic pattern transitions; clearly, including binary (or higher arity patterns) in our language would mean modifying GTran in order to ensure that changes of these kinds of pattern also force an episodic boundary.

4 Final Comments

We conclude the paper with a review of what has been presented here and a summary of further work that might be undertaken.

The contribution of the paper falls into two main parts: (i) definitions and axioms which extend the expressive power of previous work on qualitative s-t theories to handle change, continuity, rigid objects and partial knowledge and (ii) a method which exploits this theory and the heuristic of s-t inertia in order to infer qualitative s-t world models from *local surveys* using circumscription. The existence of multiple explanations is a general characteristic of abduction and even using inertia as a heuristic, many explanations will in remain general. We have discussed some possible further heuristics to prefer certain explanations, but without much more domain specific background knowledge, ambiguity will always be present. Of course, in the domain presented here of an agent exploring a world, this might provide the motivation for further exploration in order to gain more data which may then refine the possible space of explanations.

There remain many avenues for further research. Not

least of these is the implementation of the theory we have presented here and the practical evaluation of it – as Shanahan [34] has observed, efficiency *is* important, and he argues how this might be achieved in a setting not dissimilar from that presented here. Thus facing the practical challenge of dealing with the potentially very large number of possible explanations that may be present in a realistic example and developing computational methods of ensuring that these are handled efficiently is an important area for future research.

One issue that we have totally ignored is the problem of object identification over time – e.g. consider the problem of tracking a mobile object over time from video data. Of course this problem could itself be made subject to abduction – one explanation of two similarly shaped objects close to each other in time and space is that they are the same object.

Another problem mentioned by Shanahan [34] of great practical importance in the abductive scenario presented here is that of noise in the data. One advantage of using a qualitative representation is that some noise is lost in the abstraction process, though in general the problem will remain. In this context it will be useful to consider the use of qualitative languages which explicitly allow for this such as the extension of RCC to handle indeterminate boundaries [5].

In this paper we have restricted ourselves to a purely mereotopological qualitative s-t language. By increasing the expressiveness of the language by allowing other kinds of qualitative s-t knowledge (e.g. of orientation, size, distance, shape [7]) may have benefits in reducing ambiguity as multiple kinds of knowledge interact. Similarly, metric s-t knowledge may be included where available.

As already observed, knowing what kinds of behaviour patterns are (preferentially) associated with particular kinds of object or agent may help reduce the possible explanations that may be abduced. Creating suitable libraries of such behaviours is thus an important (probably domain specific) task. If there are a large number of possible such behaviours then this knowledge acquisition problem may be non trivial. In such cases, it would be useful to learn these automatically from training data, cf [11].

Another restriction made for the sake of simplicity here is that there is a single time line $(\tau_1, ..., \tau_n)$. However if we wanted to extend the theory to handle multiple cooperating local agents performing surveys asynchronously, then we would need to allow multiple time lines.

Finally, we note that the theory is potentially applicable to various other domains in which partial s-t knowledge is available and it is desirable to infer a complete scenario. One such task would be the problem of inferring what has happened between various "global snapshots" such as geographical surveys (or remote sensing data) taken at periodic intervals. Experimenting with and evaluating the approach outlined here in such contexts would also be an area worthy of further research.

Acknowledgements

The financial assistance of the EPSRC under grant GR/M56807 and EU under contract IST-2000-29375 is gratefully acknowledged as are the useful discussions within the Leeds QSR Group. We thank the anonymous referees for their valuable comments. The first author also acknowledge the Commonwealth Scholarship Commission, United Kingdom for financial assistance under reference INCS-1999-177.

References

[1] J F Allen, 'Towards a general theory of action and time', *Artificial Intelligence*, **23**(2), 123–154, (1984).

[2] B. Bennett, A. G. Cohn, P. Torrini, and S. M. Hazarika, 'Describing rigid body motions in a qualitative theory of spatial regions', in *Proc. of AAAI-2000*, pp. 503–509, (2000).

[3] B. Bennett, A. G. Cohn, P. Torrini, and S. M. Hazarika, 'A foundation for region-based qualitative geometry', in *Proc. of ECAI-2000*, pp. 204–208, (2000).

[4] A G Cohn, B Bennett, J Gooday, and N Gotts, 'RCC: A Calculus for Region based Qualitative Spatial Reasoning', *GeoInformatica*, **1**(3), 275–316, (1997).

[5] A G Cohn and N M Gotts, 'A mereological approach to representing spatial vagueness', in *Proc. of KR'96*, pp. 230–241, (1996).

[6] A G Cohn and S M Hazarika, 'Continuous transitions in mereotopology', in *Commonsense-2001: 5th Symp. on Logical Formalizations of Commonsense Reasoning*, pp. 71–80, New York, (2001).

[7] A G Cohn and S M Hazarika, 'Qualitative spatial representation and reasoning: An overview', *Fundamenta Informaticae*, **46**(1-2), 1–29, (2001).

[8] S. Coradeschi and A. Saffiotti, 'Perceptual anchoring of symbols for action', in *Proc. of IJCAI-01*, pp. 407–412, (2001).

[9] M J Egenhofer and K K Al-Taha, 'Reasoning about gradual changes of topological relationships', in *Theories and Methods of Spatio-temporal Reasoning in Geographic Space*, volume 639 of *LNCS*, pp. 196–219, (1992).

[10] M T Escrig and F Toledo, *Qualitative Spatial Reasoning: Theory and Practice - Application to Robot Navigation*, volume 47 of *Frontiers in AI and applications*, IOS Press, Amsterdam, 1998.

[11] J. H. Fernyhough, A. G. Cohn, and D. C. Hogg, 'Constructing qualitative event models automatically from video input', *Image and Vision Computing*, **18**(2), 81–103, (2000).

[12] C Freksa and K Zimmermann, 'Qualitative spatial reasoning using orientation, distance and path knowledge', *Applied Intelligence*, **6**, 49–58, (1996).

[13] A P Galton, *Qualitative Spatial Change*, Oxford University Press, 2000.

[14] P. J. Hayes, 'Naive physics I: Ontology for liquids', in *Formal Theories of the Commonsense World*, 71–89, Ablex Publ. Corp., (1985).

[15] S M Hazarika and A G Cohn, 'Qualitative spatio-temporal continuity', in *Proc. of COSIT'01*, number 2205 in LNCS, pp. 92–107, (2001).

[16] J. Hintikka, 'Model minimization - an alternative to circumscription', *Journal of Automated Reasoning*, **4**, 1–13, (1988).

[17] K Konolige, 'Belief and incompleteness', in *Formal Theories of the Commonsense World*, 359–403, Ablex Publ. Corp., (1988).

[18] B Kuipers, 'The spatial semantic hierarchy', *Artificial Intelligence*, **119**, 191–233, (2000).

[19] J-C Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, 1991.

[20] T S Levitt and D T Layto, 'Qualitative navigation for mobile robots', *Artificial Intelligence*, **44**, 305–360, (1990).

[21] V Lifschitz, 'Circumscription', in *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4, 297–352, Oxford Publ., (1994).

[22] S Lorenz, 'A tableuau prover for domain minimization', *Journal of Automated Reasoning*, **13**, 375–390, (1994).

[23] T Lozano-Pérez, 'Spatial planning: a configuration space approach', *IEEE Transactions on Computers*, **C-32**(2), 108–120, (1983).

[24] J McCarthy, 'Circumscription: A form of non-monotonic reasoning', *Artificial Intelligence*, **13**(1), 27–39, (1980).

[25] P. Muller, 'A qualitative theory of motion based on spatio-temporal primitives', in *Proc. of KR'98*, ed., A. G. Cohn et al., pp. 131–141, (1998).

[26] P. Muller, 'Space-time as a primitive for space and motion', in *Proc. of FOIS'98*, pp. 63–76, (1998).

[27] A Prescott, *Explorations in Reinforcements and Model-based Learning*, Ph.D. dissertation, University of Sheffield, 1994.

[28] D. A. Randell, Z. Cui, and A. G. Cohn, 'A spatial logic based on regions and connection', in *Proc. of KR'92*, pp. 165–176, (1992).

[29] E. Remolina and B. Kuipers, 'A logical account of causal and topological maps', in *Proc. of IJCAI-01*, pp. 5–11, (2001).

[30] E Sandewall, 'Filter preferential entailment for the logic of action in almost continuous worlds', in *Proc. of IJCAI-89*, pp. 894–899, (1989).

[31] E Sandewall, *Features and Fluents: The Representation of Knowledge about Dynamical Systems*, volume I, Oxford University Press, 1994.

[32] C Schlieder, 'Representing visible locations for qualitative navigation', in *Qualitative Reasoning and Decision Technologies*, eds., N Piera Carreté and M G Singh, pp. 523–532, Barcelona, (1993).

[33] M. Shanahan, *Solving the Frame Problem: a mathematical investigation of the common sense law of intertia*, The MIT Press, 1997.

[34] M Shanahan, 'A logical account of the common sense informatic situation for a mobile robot', *Electronic Trans. on AI*, **2**, 69–104, (1998).

[35] M. Shanahan and M Witkowski, 'High-level robot control through logic', in *Proc. of Agent Theories, Architectures and Languages*, pp. 100–113, Boston, (2000).

[36] M. Witkowski, D. Randell, and M. Shanahan, 'Deriving fluents from sensor data for mobile robots', in *Papers from the 2001 AAAI Fall Symposium*, pp. 44–51, (2001).

Building Large Composition Tables via Axiomatic Theories

David Randell **Mark Witkowski**
Department of Electrical and Electronic Engineering
Imperial College of Science, Technology and Medicine
Exhibition Road
London SW7 2BT
United Kingdom
{d.randell, m.witkowski}@ic.ac.uk

Abstract

The use of composition tables for efficiently representing and reasoning with jointly exhaustive pairwise disjoint sets of dyadic relations, is now well established in the AI literature. Whether typically built from axiomatic theories or from algebraic structures, most tables are built with a single theory in mind. We concentrate upon axiomatic theories for building these tables, and show how by factoring out related, but distinct formal theories (each capable of generating a composition table), large composition tables are easily constructed. This approach contrasts with the general difficulty of extracting out these tables where a parsimonious ontology and minimal number of primitives are used. We illustrate this with the construction of a non-trivial 20x20 composition table from two sub-theories supporting a 6x6 and 8x8 table. The ontological and representational ramifications for general theory building and the value of composition tables are discussed.

1 INTRODUCTION

The use of composition tables for efficiently representing and reasoning with Jointly Exhaustive and Pairwise Disjoint (*JEPD*) sets of n-ary relations, is now well established in AI literature. Composition tables reduce consistency checking of generated sets of ground instances of *JEPD* relations, to simple table look-up operations (Cohn, 1997). Whether typically built from axiomatic theories (e.g. Bennett, 1994; Randell, *et al.*, 1992) or algebraic structures (e.g. Egenhofer, 1994), most tables are built with a single theory in mind. But extracting these tables can sometimes be difficult, particularly in the case of the former approach, where a parsimonious ontology and minimal number of primitives are used. In this paper we use axiomatic theories as our base, but (by developing a technique first described by Galton (Galton, 1994)) show how by factoring out related, but distinct formal theories (each capable of generating a composition table) large composition tables are easily constructed. We illustrate this approach with the construction of a non-trivial 20x20 composition table from two sub-theories respectively supporting a 6x6 and 8x8 table.

In section 2 we introduce composition tables in the light of axiomatic theories, from which they can be derived. In section 3, the general method of building larger tables is described, and the formal correctness of the method given. In section 4 we present an example target axiomatic theory to which the method is applied; factor out two embedded sub-theories, then show how the composition table is generated. Section 5 gives a program outline that directly implements the method to synthesise composition tables. Section 6 discusses *conceptual neighbourhoods*, defining the valid transitions between sets of *JEPD* relations. Section 7 presents an alternative axiomatisation for a generic set of occlusion relations. Section 8 describes a program suite, incorporating a resolution based theorem prover, which we used: (i) to cross check these results, and (ii) as a generic tool to assist in the task of theory-building itself. Finally we discuss the ontological and representational ramifications for general theory building and the value of composition tables.

2 COMPOSITION TABLES

An $n \times n$ composition table takes a set n of mutually exhaustive and pair wise disjoint (*JEPD*) relations, and for each pair of relations $R_1(x,y)$ and $R_2(y,z)$, gives $R_3(x,z)$ as the set of all possible R_3 relations, implied by R_1 and R_2. In general R_3 will be a disjunction of entries; with the additional requirement that R_3 is the minimal set of instances implied by R_1 and R_2.

Composition tables are illustrated in Figures, 2, 3 and 6. Figure 6 gives the full composition table for the 20 *JEPD* relations of the Region Occlusion Calculus *ROC-20* (Randell, *et al.*, 2001), which we discuss below. This calculus is used to model occlusion relations between arbitrary shaped bodies from a given viewpoint. Figures 2 and 3 show the composition tables for two related sub-theories *RCC-8* and *ROC-6*. If $R_1(A,B)$ and $R_2(B,C)$ hold, where R_1 is the specified relation along a row, and R_2 specified down a column, the respective cell entry at the intersection encodes the complete set of values for $R_3(A,C)$. In general, each entry for the table embodies a model, and in the case where the table has been generated from an axiomatic theory, a theorem of the underlying logic. While these tables can be extracted using model building and theorem proving techniques, the extraction is not necessarily straightforward. For example, difficulties working with the spatial theory *RCC*, led to a challenge for automated theorem provers, and motivated a tractable solution by respectively encoding sub-theories of RCC into intuitionistic logic (Bennett, 1994). This has led some to pursue alternative algebraic methods using the "n-intersection" method to factor out these compositions (Egenhofer, 1994; Egenhofer *et al.*, 1994) and in a similar vein where composition tables are completely eliminated (El-Geresy and Abdelmoty, 1996).

Methods for generating composition tables currently fall into two main approaches: (i) the use of axiomatic theories, model-generation and theorem proving, and (ii) the use of algebraic structures and the intersection method. Either approach has its own particular merits. In the case of the axiomatic approach, the underlying *ontology* is highlighted and the reasoning about the domain can be applied not only to the entries of the table, but also to other *wff* that, as theorems, are not necessarily embodied in the table. For example, reasoning just using the composition table for *ROC-20* (to be described) is not sufficient to prove all the existential conditions entailed by the axiomatic theory, if completeness of the underlying theory is to be achieved. That is to say, given a model, we need to ask ourselves what set of truths are provable within our formal system. By contrast, the algebraic approach using the intersection method gains on the computational side. We argue that the method first discussed by Galton (1994) provides a practical alternative approach. While an axiomatic approach to model building is used, this is applied to sub-theories and their composition tables, where the larger theory and associated composition table is generated as a direct consequence.

Next we discuss the general method of large composition table construction, then we give an example using the theory *ROC-20* that embeds two sub-theories *RCC-8* and *ROC-6*.

3 GENERATING LARGE COMPOSITION TABLES

In his paper "Lines of Sight", Galton (1994) describes a method where by taking two independent formal theories, their composition tables, and a target set of relations constructible from these, a large composition table can be built. It is based on the assumption that each entry of the target composition table is defined in terms of the direct product of the two simpler sets of relations. We reformulate this result, by stating that given a target formal theory, one can build a composition table providing we have: (i) a set of *JEPD* definitions for the target theory, and for each sub-theory, (ii) a set of *JEPD* relations, and a composition table, and (iii) a set of *constraint axioms* that map predicates and functions defined in the target theory to predicates and functions in both sub-theories.

Let Ψ and Σ denote two sub-theories of the target theory Φ, where $JEPD^{\Psi}$, $JEPD^{\Sigma}$ and $JEPD^{\Phi}$ are respectively their *JEPD* sets of relations, and where $Rn^{\Phi}(x,y) \in JEPD^{\Phi}$; similarly: $Rn^{\Psi}(x,y) \in JEPD^{\Psi}$ and $Rn^{\Sigma}(x,y) \in JEPD^{\Sigma}$. We also assume the existence of a function ϕ, expressible in Φ, that maps between Φ and Ψ. To create the larger composition table we wish to establish each R_3 relation set for (i), assuming (ii) to (iv) shown below, where schemas (i), (ii) and (iii) respectively encode the composition tables for theories Φ, Ψ, and Σ, and where (iv) holds for each defined $Rn^{\Phi}(x,y)$ relation.

$$(i)\ R_1^{\Phi}(x,y) \,\&\, R_2^{\Phi}(y,z) \vdash R_3^{\Phi}(x,z)_1 \vee \ldots \vee R_3^{\Phi}(x,z)_n$$
$$(ii)\ R_1^{\Psi}(x,y) \,\&\, R_2^{\Psi}(y,z) \vdash R_3^{\Psi}(x,z)_1 \vee \ldots \vee R_3^{\Psi}(x,z)_n$$
$$(iii)\ R_1^{\Sigma}(x,y) \,\&\, R_2^{\Sigma}(y,z) \vdash R_3^{\Sigma}(x,z)_1 \vee \ldots \vee R_3^{\Sigma}(x,z)_n$$
$$(iv)\ Rn^{\Phi}(x,y) \leftrightarrow [Rn^{\Psi}(x,y) \,\&\, Rn^{\Sigma}(\phi(x),\phi(y))]$$

The method for generating the full, larger, composition table for Φ is as follows:

Step1: Using (iv), each single Rn^{Φ} entry is split into its two constituent relations: Rn^{Ψ} and Rn^{Σ}.

Step2: Using the composition tables encoded in (ii) and (iii); for each ordered pair $\langle R_1^n, R_2^n \rangle$ of Rn^{Ψ} and Rn^{Σ} relations, we generate two R_3^{Ψ} and R_3^{Σ} sets, that together comprise all the possible relations (as decompositions) that can be formed.

Step 3: The third step simply re-builds the set of R_3^{Φ} entries from the generated set of R_3^{Ψ} and R_3^{Σ} relations. However, typically, not all combinations formed by taking one element from each R_3^{Ψ} *and* R_3^{Σ} set will have a model. This is guaranteed by ensuring that each R_3^{Ψ} and R_3^{Σ} combination satisfies the mapping axioms (incorporating ϕ) of the theory. This leaves us with a maximal generated set of named Rn^{Φ} relations that

populate the corresponding R_3 cell in the composition table.

The correctness of the method is easily shown:

Step 1: The translation from $Rn^{\Phi}(x,y)$ to its $Rn^{\Psi}(x,y)$ and $Rn^{\Sigma}(\phi(x),\phi(y))$ constituents, is true by definition, i.e. $Rn^{\Phi}(x,y) \equiv def.\ Rn^{\Psi}(x,y)\ \&\ Rn^{\Sigma}(\phi(x),\phi(y))$.

Step 2: The two composition tables for Ψ and Σ, encoding the *theorems* of Φ guarantee that that each R_3 set of disjuncts, is both a logical consequence of R_1 and R_2, and exhaustively enumerates all the possible generated cases.

Step 3: The soundness of selecting only the $Rn^{\Psi}(x,y)$ and $Rn^{\Sigma}(\phi(x),\phi(y))$ relation pairs if $Rn^{\Phi}(x,y)$, follows immediately from the definitions and mapping axioms of the theory.

The application of steps 1-3 show that for each $R_1{}^{\Phi}$ and $R_2{}^{\Phi}$ pair, the result set $R_3{}^{\Phi}$ is both sound (i.e. is implied by the theory) and (by exhaustively generating all possible new $R_3{}^{\Phi}$ relations) is complete. **QED**.

We now develop our first-order theory to illustrate the general method.

4 ROC-20 AND RCC-8 - THE EXEMPLAR THEORIES

Our universe of discourse includes bodies, regions and points, all forming pairwise disjoint sets. A set of sorts and a *sorted logic* allowing *ad hoc* polymorphic functions and predicates to be handled, is assumed.

The notation and conventions used is as follows: *type* $a(\tau 1,..,\ \tau n)$: $\tau n+1$ means function symbol a is well sorted when its argument sorts are $\tau 1,..,\ \tau n$ with $\tau n+1$ as the result sort, and *type* $a(\tau 1,..,\ \tau n)$ means predicate a is well sorted when defined on argument sorts $\tau 1,..,\ \tau n$. Axioms, definitions and theorems are respectively indicated as follows: *(A1,..,An), (D1,..,Dn), and (T1,..,Tn)*. Where axiom/definitional schemas are used, the numbering in the parentheses reflects the number of object-level axioms and definitions generated.

We embed the mereo-topological theory *RCC-8* (Randell, *et al.*, 1992) into our theory, *ROC-20*. The same primitive dyadic relation *C/2*: '*C(x,y)*' read as "*x* is connected with *y*" is used. All the relations defined in *RCC-8* are used, and all carry their usual readings: *DC/2* (disconnected), *P/2* (part), *EQ/2* (equal), *O/2* (overlaps), *DR/2* (discrete) *PO/2* (partial overlap), *EC/2* (external connection), *PP/2* (proper part), *TPP/2* (tangential proper part), *NTPP/2* (non-tangential proper part). *PI/2*, *PPI/2*, *TPPI/2* and *NTPPI/2* are the inverse relations for *P/2*, *PP/2*, *TPP/2* and *NTPP/2*, respectively. Eight of these relations are provably *JEPD*, and are hereinafter referred to as $JEPD_{RCC\text{-}8}$.

Axioms for *C/2* and definitions for the dyadic relations of *RCC-8* are as follows:

(A1) $\forall x\ C(x,x)$
(A2) $\forall x \forall y\ [C(x,y) \rightarrow C(y,x)]$
(A3) $\forall x \forall y [\forall z[C(z,x) \leftrightarrow C(z,y)] \rightarrow EQ(x,y)]$[1]

(D1) $DC(x,y) \equiv def.\ \neg C(x,y)$
(D2) $P(x,y) \equiv def.\ \forall z[C(z,x) \rightarrow C(z,y)]$
(D3) $EQ(x,y) \equiv def.\ P(x,y)\ \&\ P(y,x)$
(D4) $O(x,y) \equiv def.\ \exists z[P(z,x)\ \&\ P(z,y)]$
(D5) $DR(x,y) \equiv def.\ \neg O(x,y)$
(D6) $PO(x,y) \equiv def.\ O(x,y)\ \&\ \neg P(x,y)\ \&\ \neg P(y,x)$
(D7) $EC(x,y) \equiv def.\ C(x,y)\ \&\ \neg O(x,y)$
(D8) $PP(x,y) \equiv def.\ P(x,y)\ \&\ \neg P(y,x)$
(D9) $TPP(x,y) \equiv def.\ PP(x,y)\ \&\ \exists z[EC(z,x)\ \&\ EC(z,y)]$
(D10) $NTPP(x,y) \equiv def.\ PP(x,y)\ \&\ \neg\exists z[EC(z,x)\ \&\ EC(z,y)]$
etc.

type $\Phi(Region,Region)$; *where* $\Phi \in$ *{C,DC,P,EQ,O,DR,PO,EC,PP,TPP,NTPP,PI,PPI, TPPI,NTPPI}*

Assumed but not given here, is an axiom in *RCC-8* that guarantees every region has a non-tangential proper part *(A3)*, and a set of axioms *(A4-A9)* introducing Boolean functions for the sum, complement, product, difference of regions, the universal spatial region; and an axiom that introduces the sort *Null* enabling partial functions to be handled – see (Randell, *et al.*, 1992) for more details.

4.1 MAPPING FUNCTIONS AND AXIOMS

ROC-20 uses *RCC-8* to model the spatial relationship between bodies, volumes, and their corresponding images with respect to a viewpoint. The formal distinction is maintained by introducing two functions: '*region(x)*' read as "the region occupied by *x*" and '*image(x,v)*' read as "the image of *x* with respect to viewpoint *v*". The function: *region/1*, maps a body to the volume of space it occupies, and *image/2* maps a body and a viewpoint to its image; i.e. the region defined by the set of projected half-lines originating at the viewpoint and intersecting the body, so forming part of the surface of a sphere of infinite radius centred on the viewpoint. A set of axioms acting as

[1] Strictly speaking axiom *(A3)* is immediate consequence of definitions *(D2)* and *(D3)*, but is added here simply to clarify the relationship between the relations *C/2* and *EQ/2*.

a set of *spatial constraints* between bodies, a given viewpoint, and their corresponding images are given[2]:

(A11-A15) $\forall x \forall y\ [\Phi(region(x),region(y)) \rightarrow \forall v\ [\Phi(image(x,v),image(y,v))]]$

type *region(Body):Region*[3]
type *image(Body,Point):Region*
type Φ*(Region,Region) where:*
$\Phi \in \{C,O,P,NTPP,EQ\}$[4]

4.2 OCCLUSION

For the occlusion part of the theory, a second primitive relation: '*TotallyOccludes(x,y,v)*', read as "*x* totally occludes *y* with respect to viewpoint *v*", is introduced. *Totally Occludes/3* is axiomatised to be transitive. Several other axioms are used to embed *RCC-8* into this theory, making *TotallyOccludes/3* additionally asymmetrical and irreflexive.

The intended *geometric* meaning of total occlusion is as follows. Let *line(p1,p2,p3)* mean that points *p1*, *p2* and *p3* fall on a straight line with *p2* strictly between *p1* and *p3*. Then, *x* totally occludes *y* from *v iff* for every point *p* in y, there exists a point *q* in *x* such that *line(v,q,p)*, and there are no points *p'* in *y*, and *q'* in *x*, such that *line(v,p',q')*. An object *x* can totally occlude an object *y* even if *x* itself is totally occluded by another object.

Axiom *A16* (below) states that if *x* totally occludes *y*, *x* totally occludes any part of *y*; and *A17* if *x* totally occludes *y* no part of *y* totally occludes part of *x*. *A17* excludes cases of total occlusion where part of the occluding wraps 'behind' the occluded object. This is an example of *mutual occlusion*, and which is defined below in definition (*D17*). *A18* states that if *x* totally occludes *y*, the image of *x* subtends the image of *y*. Note that *A18* is not a biconditional because the *P/2* relation (defined on images here) is indifferent to various factors including relative distance and overlap between occluding bodies in the assumed model. Spatial identity of regions in terms of co-location still applies, but is restricted to the dimensionality of the regions being modelled.

(A16) $\forall x \forall y \forall z \forall v\ [[TotallyOccludes(x,y,v) \ \&\ TotallyOccludes(y,z,v)] \rightarrow TotallyOccludes(x,z,v)]$

(A17) $\forall x \forall y \forall v\ [TotallyOccludes(x,y,v) \rightarrow \forall z u[[P(region(z),region(x)) \ \&\ P(region(u),region(y))] \rightarrow \neg TotallyOccludes(u,z,v)]]$

(A18) $\forall x \forall y \forall v\ [TotallyOccludes(x,y,v) \rightarrow P(image(y,v),image(x,v))]$

type *TotallyOccludes(Body,Body,Point)*

Total occlusion between distinct bodies implies occlusion (*T1*), which in turn implies region overlap between their corresponding images (*T2*). Moreover, we can also show that if *x* totally occludes *y*, *x* totally occludes every part of *y* (*T3*) and that from every viewpoint, body *x* has two parts (*y* and *z*) such that the one part (*y*) totally occludes the other part (z). This can be interpreted in a 3D model to mean that bodies have *depth* (*T4*):

(T1) $\forall x \forall y \forall v[TotallyOccludes(x,y,v) \rightarrow Occludes(x,y,v)]$

(T2) $\forall x \forall y \forall v\ [Occludes(x,y,v) \rightarrow O(image(x,v),image(y,v))]$

(T3) $\forall x \forall y \forall v[TotallyOccludes(x,y,v) \rightarrow \forall z[P(region(z),region(x))] \rightarrow TotallyOccludes(x,z,v)]$

(T4) $\forall x \forall v \exists y \exists z[P(region(y),region(x)) \ \&\ P(region(z),region(x)) \ \&\ TotallyOccludes(x,y,v)]$

A refined set of occlusion relations is defined including weak occlusion, and partial and mutual occlusion. '*Occludes(x,y,v)*' is read as "*x* occludes *y* from viewpoint *v*" and means from *v* that some part of *x* totally occludes some part of *y*. *Occludes/3* in contrast to *O/2* is *non-symmetrical*. Other more specific occlusion relations are defined and then mapped to their *RCC-8* analogues. For completeness (not listed here) inverse relations are given for *Occludes/3*, *TotallyOccludes/3* and *PartiallyOccludes/3* (*D18-D20*); leaving the null case: *NonOccludes/3*, where no occlusion arises. The six relations: *NonOccludes/3*, *MutuallyOccludes/3*; and *TotallyOccludes/3*, *PartiallyOccludes/3*, and their inverses also form another *JEPD* set - $JEPD^{ROC\text{-}6}$. For more details see (Randell *et al.*, 2001).

(D15) $Occludes(x,y,v) \equiv def.\ \exists z \exists u[P(region(z),region(x)) \ \&\ P(region(u),region(y)) \ \&\ TotallyOccludes(z,u,v)]$

---

[2] Although not developed here, the distinction made between bodies and regions enables one to define the notion of free space and model spatial occupancy – see (Shanahan, 1996).

[3] Sortal declarations given here are not as restricted as they could be, for example we could declare: ***type*** *region(Body): 3DRegion*, and ***type*** *image(Body,Point): 2DRegion*, where *2DRegion* and *3DRegion* are (disjoint) subsorts of the sort *Region*.

[4] The set of predicate constants used here (and in the set of axioms for *ROC-20* that appear in this paper) differs from that presented in (Randell *et al.*, 2001) where redundancy in the original set of axioms has been addressed. The exception is the removal of axiom (*A13*) in that paper, for which counter-examples have been found. We wish to thank Antony Galton for bringing our attention to this.

(D16) PartiallyOccludes(x,y,v) ≡def.
Occludes(x,y,v) &
¬TotallyOccludes(x,y,v) &
¬Occludes(y,x,v)
(D17) MutuallyOccludes(x,y,v) ≡def.
Occludes(x,y,v) & Occludes(y,x,v)
(D21) NonOccludes(x,y,v) ≡def.
¬Occludes(x,y,v) & ¬Occludes(y,x,v)

(A19) ∀x∀y∀v[NonOccludes(x,y,v) →
DR(image(x,v),image(y,v))]
(A20) ∀x∀y∀v[PartiallyOccludes(x,y,v) →
[PO(image(x,v),image(y,v)) ∨
PP(image(x,v),image(y,v))]]
(A21) ∀x∀y∀v[MutuallyOccludes(x,y,v) →
O(image(x,v),image(y,v))

type *Φ(Body,Body,Point)*; where *Φ ∈ {Occludes, PartiallyOccludes, MutuallyOccludes, NonOccludes, ...}*

Finally, a total set of 20 *JEPD* relations, $JEPD^{ROC\text{-}20}$ is defined using more specific instances on the *P/2* relation. These are generated using the following definitional schemas, and are illustrated with a graphical model shown in Figure 1. This also provides a key to the 5x4 matrices that populate the cell entries for the 20x20 composition table illustrated in Figure 6. In each case a filled/unfilled square, respectively indicates a model/no model. This then completes the development of the basic theory that is sufficient for our purposes here.

(D22-D33) ΦΨ(x,y,v) ≡def.
Φ(x,y,v) & Ψ(image(x,v),image(y,v))
(D34-D41) $X\Psi^{-1}(x,y,v) \equiv_{def.}$
X(y,x,v) & Ψ(image(y,v),image(x,v))

type *ΦΨ(Body,Body,Point)*
type $X\Psi^{-1}$*(Body,Body,Point)*
type *Φ(Body,Body,Point)*
type *X(Body,Body,Point)*
type *Ψ(Region,Region)*

where if:

Φ = NonOccludes, then Ψ ∈ {DC,EC}
Φ = PartiallyOccludes, then Ψ ∈ {PO,TPP,NTPP}
Φ = TotallyOccludes, then Ψ ∈ {EQ,TPPI,NTPPI}
Φ = MutuallyOccludes, then Ψ ∈ {PO,EQ,TPP,NTPP}

and where if:

X = PartiallyOccludes, then Ψ ∈{PO,TPP,NTPP}
X = TotallyOccludes, then Ψ ∈ {EQ,TPPI,NTPPI}
X = MutuallyOccludes, then Ψ ∈ {TPP,NTPP}

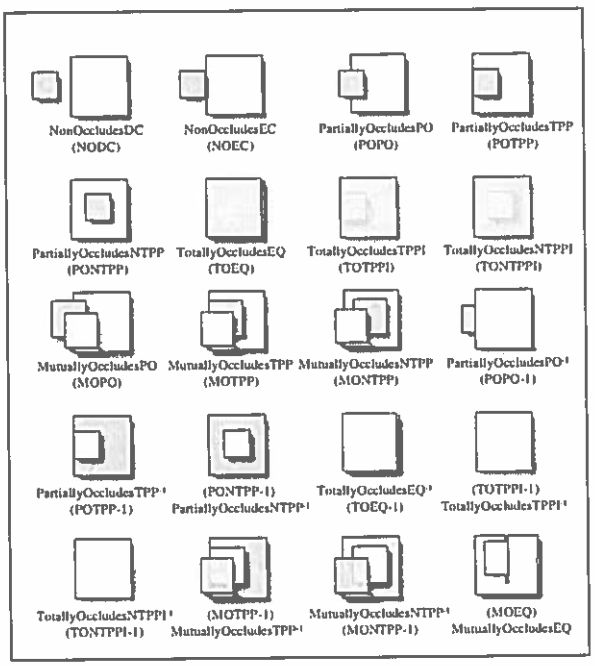


Figure 1: Graphical model for the *ROC-20* relations.

5 GENERATING THE 20X20 COMPOSITION TABLE

The method used to compute the *20×20* composition table exploits that outlined by Galton. In our case it is based on the fact that each *ROC-20* relation can be defined in terms of the direct product of two simpler (i.e. *ROC-6* and *RCC-8*) sets of relations; the composition table for the product set being directly generated from the composition tables of its factors. Where *RCC-8* considers modes of connection between regions, *ROC-6* considers generic occlusion relationships between bodies with respect to a viewpoint.

The composition table for *ROC-6* is shown in Figure 3, the relation names: NO, PO, TO, MO, POI and TOI, respectively abbreviate: *NonOccludes, PartiallyOccludes, TotallyOccludes, MutuallyOccludes*, and the two inverse relations: $PartiallyOccludes^{-1}$ and $TotallyOccludes^{-1}$.

	DC	EC	PO	TPP	TPPI	EQ	NTPP	NTPPI
DC	DC EC PO TPP TPPI EQ NTPP NTPPI	DC EC PO TPP NTPP	DC EC PO TPP NTPP	DC EC PO TPP NTPP	DC	DC	DC EC PO TPP NTPP	DC
EC	DC EC PO TPPI NTPPI	DC EC PO TPP TPPI EQ	DC EC PO TPP NTPP	EC PO TPP NTPP	DC EC	EC	PO TPP NTPP	DC
PO	DC EC PO TPPI NTPPI	DC EC PO TPPI NTPPI	DC EC PO TPP TPPI EQ NTPP NTPPI	PO TPP NTPP	DC EC PO TPPI NTPPI	PO	PO TPP NTPP	DC EC PO TPPI NTPPI
TPP	DC	DC EC	DC EC PO TPP NTPP	TPP NTPP	DC EC PO TPP TPPI EQ	TPP	NTPP	DC EC PO TPPI NTPPI
TPPI	DC EC PO TPPI NTPPI	EC PO TPPI NTPPI	PO TPPI NTPPI	PO TPP TPPI EQ	TPPI NTPPI	TPPI	PO TPP NTPP	NTPPI
EQ	DC	EC	PO	TPP	TPPI	EQ	NTPP	NTPPI
NTPP	DC	DC	DC EC PO TPP NTPP	NTPP	DC EC PO TPP NTPP	NTPP	NTPP	DC EC PO TPP TPPI EQ NTPP NTPPI
NTPPI	DC EC PO TPPI NTPPI	PO TPPI NTPPI	PO TPPI NTPPI	PO TPPI NTPPI	NTPPI	NTPPI	PO TPP TPPI EQ NTPP NTPPI	NTPPI

Figure 2: Composition table for *RCC-8*

	NO	PO	POI	TO	MO	TOI
NO	NO PO POI TO MO TOI	NO PO POI MO TOI	NO PO POI MO TOI	NO	NO PO POI MO TOI	NO PO POI MO TOI
PO	NO PO POI TO MO	NO PO POI TO MO	NO PO POI TO MO TOI	NO PO TO	NO PO POI TO MO TOI	PO POI MO TOI
POI	NO PO POI TO MO	NO PO POI TO MO TOI	NO PO POI MO TOI	NO PO POI TO MO	NO PO POI MO TOI	POI MO TOI
TO	NO PO POI TO MO	PO TO MO	PO POI TO MO	TO	PO TO MO	PO POI TO MO TOI
MO	NO PO POI TO MO	NO PO POI TO MO	NO PO POI TO MO TOI	NO PO POI TO MO	NO PO POI TO MO TOI	POI MO TOI
TOI	NO	NO PO POI MO TOI	NO POI TOI	NO PO POI TO MO TOI	NO PO POI MO TOI	TOI

Figure 3: Composition table for *ROC-6*

We established the relative consistency and correctness of the composition table entries for *ROC-6* by interpreting the relations against a graphical model that satisfied the axioms of the theory. That is to say a handcrafted model was generated for each of the (156) individual R_1^6, R_2^6 and R_3^6 cell entries. This result was then independently checked against the output from a batch program that used the resolution theorem prover *SPASS* - requiring a total of $6^3 = 216$ potential theorems to check (Section 8). In the case of *RCC-8*, the composition table was taken as a given – see Bennett (1994).

5.1 A WORKED EXAMPLE

Let: $Rn^{20}(x,y,v) \in JEPD^{ROC\text{-}20}$ be an instance of $Rn^{\Phi}(x,y) \in JEPD^{\Phi}$; similarly let: $Rn^6(x,y,v) \in JEPD^{ROC\text{-}6}$ be an instance of $Rn^{\Psi}(x,y) \in JEPD^{\Psi}$ and $Rn^8(x,y) \in JEPD^{RCC\text{-}8}$ of $Rn^{\Sigma}(x,y) \in JEPD^{\Sigma}$. The function *image/2* substitutes for ϕ. Therefore, we wish to establish each R_3^{20} relation set for (i), assuming (ii) to (iv):

(i) $R_1^{20}(x,y,v) \;\&\; R_2^{20}(y,z,v) \vdash$
$\quad R_3^{20}(x,z,v)_1 \vee \ldots \vee R_3^{20}(x,z,v)_n$
(ii) $R_1^6(x,y,v) \;\&\; R_2^6(y,z,v) \vdash R_3^6(x,y,v)_1 \vee \ldots \vee R_3^6(x,z,v)_n$
(iii) $R_1^8(x,y) \;\&\; R_2^8(y,z) \vdash R_3^8(x,z)_1 \vee \ldots \vee R_3^8(x,z)_n$
(iv) $Rn^{20}(x,y,v) \leftrightarrow$
$\quad [Rn^6(x,y,v) \;\&\; Rn^8(image(x,v),image(y,v))]$

where, as before, the schemas (i), (ii) and (iii) respectively encode the composition tables for *ROC-20* (Φ), *ROC-6* (Ψ), and *RCC-8* (Σ), where (iv) holds for each defined $Rn^{20}(x,y,v)$ relation. In the case of *ROC-20* and *ROC-6* each *JEPD* relation is ternary, but where the last argument acts only as an index term.

In the following example, a cell entry for *ROC-20's* composition table is derived and justified against the theory. The three steps highlighted mirror those given in section 3. We wish to compute the cell entries for *R3*, where:

$R1 = R_1^{20} = TotallyOccludesTPPI(a,b,v1)$
$R2 = R_2^{20} = PartiallyOccludesPO(b,c,v1)$

Step 1: Unpack the clauses using the definitions.

R1 = {TotallyOccludes(a,b,v1),
TPPI(image(a,b,v1),image(b,c,v1))}
R2 = {PartiallyOccludes(b,c,v1),
PO(image(b,c,v1),image(b,c,v1))}

Step 2: Compute the composition table entries (atoms here) for each paired set of relations:

<TotallyOccludes(a,b,v), PartiallyOccludes(b,c,v1)> =
{PartiallyOccludes(a,c,v1),
TotallyOccludes(a,c,v1),
MutuallyOccludes(a,c,v1)}

<TPPI(image(a,b,v1),image(b,c,v1)),
PO(image(b,c,v1),image(b,c,v1))> =
{PO(image(a,v1),image(c,v1)),
TPPI(image(a,v1),image(c,v1)),
NTPPI(image(a,v1),image(c,v1))}

Step 3: Take the cross-product between both sets of generated atoms, and rebuild those satisfying the definitions and any other constraints arising from the underlying theory:

(A22) $\forall x \forall y \forall v$ *[TotallyOccludes(x,y,v)* $\rightarrow$
P(image(y,v),image(x,v))]

rules out:

{TotallyOccludes(a,c,v1),PO(image(a,v1),image(c,v1))},

owing to the theorem:

$$\forall x \forall y\ [PI(x,y) \leftrightarrow [EQ(x,y) \vee TPPI(x,y) \vee NTPPI(x,y)]],$$

(A24) $\forall x \forall y \forall v[PartiallyOccludes(x,y,v) \rightarrow$
[PO(image(x,v),image(y,v)) ∨
PP(image(x,v),image(y,v))]]

rules out:

{PartiallyOccludes(a,c,v1),TPPI(image(a,v1),image(c,v1))},
{PartiallyOccludes(a,c,v1),NTPPI(image(a,v1),image(c,v1))},

owing to the theorem:

$$\forall x \forall y\ [PP(x,y) \leftrightarrow [TPP(x,y) \vee NTPP(x,y)]]$$

(A25) $\forall x \forall y \forall v[MutuallyOccludes(x,y,v) \rightarrow$
[PO(image(x,v),image(y,v)) ∨
P(image(x,v),image(y,v)) ∨
PI(image(x,v),image(y,v))]]

no cases ruled out.

Hence: R3 = {PartiallyOccludesPO(a,c,v,1),
TotallyOccludesTPPI(a,c,v,1),
TotallyOccludesNTPPI(a,c,v,1),
MutuallyOccludesPO(a,c,v1),
MutuallyOccludesTPP^{-1}(a,c,v1),
MutuallyOccludesNTPP^{-1}(a,c,v1)}

5.2 THE PROGRAM

The full *ROC-20* composition table (including the graphical output) shown in Figure 6 was automatically generated using a program that implements the method described in section 3. Each 5x4 matrix (as the intersection of a row and column) encodes a R_3^{20} disjunctive entry, for a given R_1^{20} and R_2^{20} pair. A filled/unfilled square respectively represents a single R_3^{20} disjunction that has a model/no model. The 20x20 composition table results were also confirmed using the resolution theorem prover *SPASS* [http://spass.mpi-sb.mpg.de/] – which is discussed further below.

Let `tab1`, `tab2` and `tab3` each be three-dimensional arrays encoding, respectively, the two smaller composition tables, Ψ (*ROC-6*) and Σ (*RCC-8*), and the larger composition table Φ (*ROC-20*) to be synthesised. The first and second indices select a cell by row and column, the third an element in a cell. The size of each table will be `size1`, `size2` and `size3`.

Let `deftab` be the table of Rn^{Φ} entries defining the constituent pairings between the two smaller theories (Rn^{Ψ} and Rn^{Σ}). By definition the dimensions of `deftab` are [`size3`,2]. From section 4.3 `deftab` is initialised thus:

```
deftab := {{NO,DC}, {NO,EC}, {PO,PO}, {PO,TPP},
  {PO,NTPP}, {TO,EQ}, {TO,TPPI}, {TO,NTPPI},
  {MO,RPO}, {MO,TPP}, {MO,NTPP}, {POI,PO},
  {POI,TPPI}, {POI,NTPPI}, {TOI,EQ}, {TOI,TPP},
  {TOI,NTPP}, {MO,TPPI}, {MO,NTPPI}, {MO,EQ}};
```

This corresponds to step 1 of section 3. (Note the use of program constants `POI` and `TOI` to denote the inverse relations defined in *D34-41*.)

The psuedo-code for the table synthesis procedure is given below:

```
BuildTableBySynthesis(deftab, tab1, tab2, tab3,
                                  size1, size2, size3)
  {
  clear(tab3);
  for (i := 1 to size3)
    {
    for (j := 1 to size3)
      {
      cell1[] := tab1[deftab[i,1],deftab[j,1]];
      cell2[] := tab2[deftab[i,2],deftab[j,2]];
      for (k := 1 to size1)
        {
        for(l := 1 to size2)
          {
          for (m := 1 to size3)
            {
            if (cell1[k] = deftab[m,1] AND
                cell2[l] = deftab[m,2])
              {
              tab3[i,j,m] := 1;
              }
            }
          }
        }
      }
    }
  }
```

The loops (`i`, `j`, `k` and `l`) sequentially generate every possible ordered pair $<R_1^n,R_2^n>$ (step 2 of section 3). The innermost loop (`m`) checks whether each combination so generated satisfies the mapping axioms (as encoded in `deftab`) and populates the equivalent entry in the large composition table `tab3` if it is satisfied (step 3).

6 CONCEPTUAL NEIGHBOURHOODS

An inspection of the composition tables for *ROC-6* and *ROC-20*, shows that each R_3 entry forms a *conceptual neighbourhood* (Freksa, 1992), i.e. each set of elements forms a connected subset of relations for each corresponding neighbourhood diagram.

A neighbourhood diagram for *ROC-20* reworked as an *envisionment table* is given in (Randell *et al.*, 2001), but this is easily generated from the map between the two neighbourhood diagrams for $JEPD^{RCC-8}$ and $JEPD^{ROC-6}$ as follows. Let: $Rn^{20}(x,y,v) \in JEPD^{ROC-20}$; $Rn^{6}(x,y,v) \in$

$JEPD^{ROC\text{-}6}$ and $Rn^8(x,y) \in JEPD^{RCC\text{-}8}$; where for each Rn^{20} definition: $Rn^{20}(x,y,v) \leftrightarrow [Rn^6(x,y,v) \ \& \ Rn^8(image(x,v),image(y,v))]$. Then given the pair of relations: $(R_1^{20}(x,y,v), R_2^{20}(x,y,v))$ a path exists in the neighbourhood diagram *for ROC-20 iff* a neighbourhood path exists between: $(R_1^8(x,y), R_2^8(x,y))$ *and* $(R_1^6(x,y,v), R_2^6(x,y,v))$.

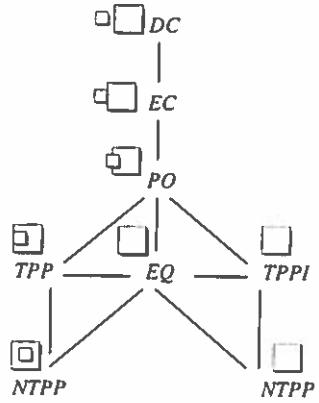


Figure 4: Neighbourhood diagram for RCC-8

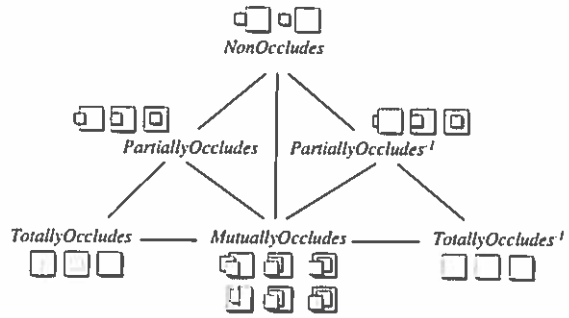


Figure 5: Neighbourhood diagram for ROC-6

7 ROC-6 AND RCC-5

Given the fact that *ROC-6* does not use any *RCC-8* relation greater than the part-whole relation *P/2*, a formal relationship between *ROC-6* and the weaker mereo-topological theory *RCC-5* (Bennett, 1994) is naturally suggested. However, even though we can axiomatise *RCC-5*, the formal relationship between the two is not a simple matter of subsumption.

To axiomatise *RCC-5*, the stronger primitive dyadic overlap relation *O/2* (c.f. *C/2* for *RCC-8*) is used. '*O(x,y)*' is read as "*x* overlaps *y*", meaning that a region is shared in common. Many of the relations named in *RCC-8* are used, and all carry the same readings and sortal declarations: *P/2* (part), *EQ/2* (equal), *DR/2* (discrete) *PO/2* (partial overlap), *PP/2* (proper part), with the inverse relations *PI/2, and PPI/2*. Of these, five (i.e. *PO/2, PP/2,EQ/2, PP-1/2*, and *DR/2*) are *JEPD*.

Axioms for *O/2*, definitions for the dyadic relations of *RCC-5*, and the mapping axioms are now restricted to the predicates {O,P,EQ}, which follow those for *ROC-20*. The exception is the P/2 relation, which is now defined directly in terms of *O/2* and not *C/2* as before:

(A1') $\forall x\, O(x,x)$
(A2') $\forall x \forall y\, [O(x,y) \rightarrow O(y,x)]$
(A3') $\forall x \forall y [\forall z [O(z,x) \leftrightarrow O(z,y)] \rightarrow EQ(x,y)]$

(D2') $P(x,y) \equiv def.\ \forall z [O(z,x) \rightarrow O(z,y)]$
etc.

Again mirroring *RCC-8*, but not reproduced here, is an axiom in *RCC-5* that guarantees every region has a proper part, and a set of axioms replicating the same Boolean functions used in *RCC-8*. The only difference between the axioms used in *RCC-8* compared with their analogues in *RCC-5*, is that '*C(x,y)*' appearing in clauses in the axiomatisation for *RCC-8* is substituted with '*O(x,y)*'; with the single exception of the complement function *compl(x)*. This changes from: $\forall x \forall y [[C(x,compl(y)) \leftrightarrow \neg NTPP(x,y)] \ \& \ [O(x,compl(y)) \leftrightarrow \neg P(x,y)]]$, to: $\forall x \forall y\, [O(x,compl(y)) \leftrightarrow \neg P(x,y)]$. This change has an important formal consequence that is discussed below. From here on, we simply mirror axioms (A16-21), definitions (D15-21), and the mapping axioms described in section 4.1, but restricted to the set {*O,P,EQ*} governing the $JEPD^{ROC\text{-}6}$ set of relations, yielding the theory *ROC-6*.

One cannot simply reduce *RCC-8*, as presented in (Randell, *et al.*, 1992), to *RCC-5* by the simple addition of the 'reduction' axiom: $\forall x \forall y [C(x,y) \rightarrow O(x,y)]$, (i.e. meaning external connection between regions is not allowed in the domain) without contradiction. The restriction stems from the complement axiom:

$$\forall x \forall y [[C(x,compl(y)) \leftrightarrow \neg NTPP(x,y)] \ \& \ [O(x,compl(y)) \leftrightarrow \neg P(x,y)]].$$

With *C/2* stipulated to be equivalent to *O/2*, *TPP/2* can never be true in the domain, and *PP/2* becomes equivalent to *NTPP/2*; meaning $\forall x [O(x,compl(x)) \leftrightarrow \neg PP(x,x)]$ follows. But given $\forall x \neg PP(x,x)$ is true, each region overlaps (i.e. shares a region in common with) its complement – which leads to the contradiction.

8 THEORY CORROBORATION

In addition to computing the 20x20 entries using the program described in section 5.2 we also confirmed the results against output provided by the resolution theorem proving program *SPASS*. In both cases each of the *JEPD* relations were encoded as bitmaps; while in the case of the latter, a customised shell program was developed that interfaced to the theorem prover functioning as a flexible, general purpose *theory development tool*.

First, all the axioms and definitions of *ROC-20* were coded up in SPASS notation. Then the program steps were as follows. (i) We incrementally checked the axiom set for possible redundancy, by trying to prove each axiom in turn as a theorem. Axioms in the set that proved to be redundant were simply flagged. (ii) We automatically generated a set of clauses (in SPASS notation) encoding the set of $JEPD^{ROC\text{-}6}$ relations and tested that they were indeed, jointly exhaustive and pairwise disjoint. These clauses were then appended to the original axiom set, solely for the purposes of program efficiency. (iii) A set of $(6^3 = 216)$ clauses which encoded potential theorems for each composition table entry were similarly machine generated and batch processed, and on completion were again added to the SPASS axiom set. This completed the run for *ROC-6*. For *ROC-20* the same steps were repeated: (iv) first testing that the set of $JEPD^{ROC\text{-}20}$ relations were *JEPD*, and finally (v) automatically constructing each composition table entry for *ROC-20* requiring 20^3 (8,000) potential theorems to be checked. In order to reduce the overall run time, the time-out allocated for the individual clauses to be proved was weighted in favour of predicted theorems. This was determined by the results generated by the 'synthesised composition table generation' method.

Despite the obvious limitation imposed by the implemented logic, i.e. using a *semi-decidable system* (meaning no practical distinction can be made between distinguishing between non-theorems, or theorems not yet proved) the program nevertheless threw up several interesting surprises. The first was revealing redundancy in several *ROC-20* axioms that had hitherto escaped our attention. This was despite an increasing familiarity we were building up with the developing theory. This pruning, in turn, helped us to identify a simpler abstract graphical model for the theory. Secondly, after only a short period into a few program runs, we soon noticed anomalous results that could not be accounted for in our model. On closer inspection the anomaly was quickly traced to an incorrect data entry error in the composition table for *RCC-8*. That this appeared very quickly, highlighted two things about this approach: (i) by virtue of the modular nature of the shell program, the correction was easily done, leaving only background processing time for a complete program re-run; and (ii) increasing confidence in the formal correctness of the developing theory was gained, as more clauses were incrementally added without showing any sign of a problem on the interpreted output. Finally, it is noteworthy that when using an earlier set of relations, a simple *JEPD* test failed to prove the relation set was exhaustive. This pointed to the existence of a 'missing' node (*MOEQ/2*) in the embedding relational lattice and that too strong a model had been used to interpret the theory; which indeed proved to be the case. In each case the flexibility of the shell program has proved to be a very useful tool for the process of theory development.

9 WORKING METHODOLOGY AND ONTOLOGICAL ISSUES.

This approach described here, gains over the naïve method of generating a large composition table from a single unified theory in several ways. Firstly, by factoring out sub-theories, it clarifies the underlying ontology, and how these sub-theories map between each other. Secondly, with the theoretical and practical need to compare theories, the modular approach allows existing theories to be re-used rather than building yet another axiomatic theory whose only connection with an existing theory may be implicit from the assumed model and informal semantics given.

There are two approaches we can take. Top down, we take a single theory, identify and factor out subsets of n-ary *JEPD* relations, embed these in relational lattices and extract out smaller compositions tables for each. In contrast, building bottom up, while we can generate the composition table for the larger theory from two (or more) sub-theories, the consequence classes for both will not be necessarily identical. While some reasoning tasks using composition tables are adequate, e.g. satisfiability checking of sets of atomic propositions, not all are. Clearly, composition tables (encoding a set of *universal axioms*) cannot capture all the *existential conditions* of the underlying theory; but neither do they encode all the universal relational properties of the defined predicates. For example, the set of universal axioms used to define the composition table for *RCC-8* and those encoding their *JEPD* properties, is not sufficient to prove the symmetry of the relations *C/2* and *O/2*.

10 CONCLUSIONS AND FUTURE WORK

An open question remains about how much of the above method can be automated. Currently, we hand-build our target set of defined *JEPD* relations, but it is certainly possible to machine generate these given a weaker *JEPD* superset of defined relations, and then for each pair of

relations from that set, automatically checking subsumption and pairwise disjoint relationships. Currently we use a customised implementation of the resolution refutation program SPASS that takes an existing axiomatic theory, and automatically builds and checks lemmas (specifically *JEPD* and composition table information) and redundancy of the axioms used. This concentrates specifically upon the task of building, as opposed to implementing and *using* such theories in an applied domain, e.g. using *ROC-20* in a real-world Cognitive Robotics programme.

One promising approach arises simply from being able to factor out and map between identified sub-theories and their respective composition tables. This metalogical structure and decomposition naturally leads into parallel or hierarchical search techniques. For example, using *ROC-20*, we can map *wff* defined in *ROC-20* (modelling bodies) to *wff* defined in *RCC-8* (their corresponding images), restricting the refutation search to a hierarchically and increasingly constrained set of *RCC-8*, *ROC-6* or *ROC-20* formulae, or alternatively to conduct the search in parallel.

From a theoretical standpoint, more work is required to see exactly what the conceptual neighbourhood property gives with respect to the task of showing your axiomatic theory is both correct and complete, for the intended domain (c.f. Duentsch *et al.*, 1998).

The method illustrated here, is very general and can be easily extended to build very large formal theories by mapping between two or more sub-theories, each of which gives rise to a set of *JEPD* relations and their associated composition table. Axioms that map between predicates and functions between the theories are singled out, and used as (but also serve to partition the theory into) sub-theories and the map between them.

Acknowledgements

Work described here has been supported by EPSRC project GR/N13104, "Cognitive Robotics II". We wish to thank the anonymous referees for their detailed comments and suggestions. We would also like to thank Antony Galton, Brandon Bennett, Baher El-Geresy, Paulo Santos and Murray Shanahan for useful comments and feedback on the material used in this paper.

References

Bennett, B. (1994) Spatial Reasoning with Propositional Logic, *Proc. 4th Int. Conf. on Knowledge Representation and Reasoning (KR-94)*, pages 51-62

Cohn, A. G. (1997) Qualitative spatial representation and reasoning techniques, *Proc. KI-97, Advances in Artificial Intelligence*, LNAI 1303, Springer, pages 1-30

Duentsch, I., Wang, H., McCloskey, S. (1998) A Relation-algebra Approach to the Region Connection Calculus, Technical Report, School of Information and Software Engineering, University of Ulster.

Egenhofer, M. (1994) Deriving the Composition of Binary Topological Relations. *J. of Visual Languages and Computing*, Vol. 5, pages 133-149

Egenhofer, M., Clementini, E. and Di Felice, P. (1994) Topological Relations Between Regions with Holes, *Int. J. of Geological Information Systems*, Vol. 8-2, pages 129-144

El-Geresy, B. and Abdelmoty, A. (1996) Order in Space: A General Formalism for Spatial Reasoning, in: *Proc. 8th Int. Conf. on Tools with Artificial Intelligence (TAI'96)*, pages 183-193

Freksa, C. (1992) Conceptual Neighborhood and its Role in Temporal and Spatial Reasoning, in: Singh, M., Trave-Massuyes, L. (eds.) Decision Support Systems and Qualitative Reasoning, pages 181-187, North-Holland, Amsterdam

Galton, A. P. (1994) Lines of Sight, *AISB Workshop on Spatial and Spatio-Temporal Reasoning*.

Randell, D. A., Cui, Z. and Cohn A. G. (1992) A Spatial Logic Based on Regions and Connection, *Proc. 3rd Int. Conf. on Knowledge Representation and Reasoning (KR-92)*, Morgan Kaufmann, San Mateo, pages 165-176

Randell, D., Witkowski, M. and Shanahan, M. (2001) From Images to Bodies: Modelling Spatial Occlusion and Motion Parallax, *Proc 17th Int. Joint Conf. on Artificial Intelligence*, Morgan Kaufmann, Seattle, pages 57-63

Shanahan, M. P. (1996) Robotics and the Common Sense Informatic Situation, *Proc. 12th Euro. Conf. on Artificial Intelligence (ECAI-96)*, pages 684-688

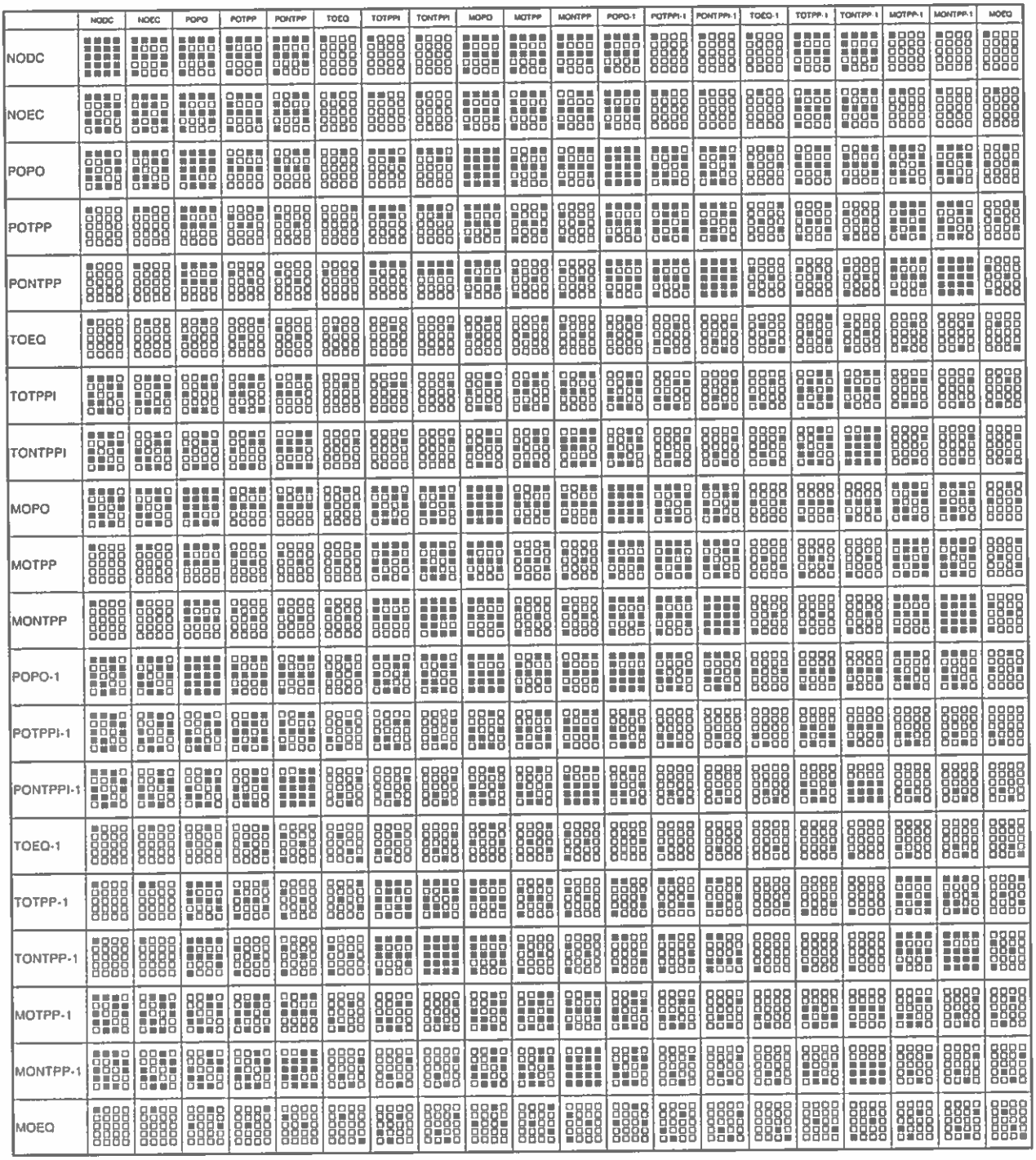


Figure 6: Full Composition Table for *ROC-20* - see Figure 1 for explanatory key

Design and comparison of lattices of topological relations based on Galois lattice theory

Florence Le Ber and Amedeo Napoli
LORIA UMR 7503
BP 239, 54506 Vandœuvre-lès-Nancy, France
leber@loria.fr, napoli@loria.fr

Abstract

This paper presents an approach to spatial representation and reasoning with lattices of topological relations. This approach is based on the Galois lattice theory that is used to create *concepts*, by associating sets of objects to sets of attributes. The objects considered here are the base relations of the Region Connection Calculus **RCC-8** and the attributes are computational conditions that are used in geographic information systems for modeling topological relations. Three lattices, named $\mathcal{T}_{\mathcal{GE}}$, $\mathcal{T}_{\mathcal{GL}}$, and $\mathcal{T}_{\mathcal{GM}}$, involving three sets of computational conditions, are introduced. The design and the properties of these lattices are detailed and then compared on the basis of the underlying computational conditions. Finally the three lattices are compared to each others, and to a fourth lattice, $\mathcal{T}_{\mathcal{P}}$, that is based on the set of all disjunctions of **RCC-8** relations, with respect to qualitative spatial reasoning capabilities.

1 INTRODUCTION

This paper presents an original approach to spatial representation and reasoning with topological relations based on the use of lattices of relations. The topological relations considered here are those of the Region Connection Calculus **RCC-8** [Cohn et al., 1997, Randell et al., 1992]. Three different lattices of topological relations are introduced and compared. They are built thanks to the Galois connection, that is used to create *concepts*, by associating sets of objects (the *extents* of the concepts) to sets of attributes (the *intents*)[1] [Barbut and Monjardet, 1970, Davey and Priestley, 1990]. These lattices allow the representation of topological relations both with computational operations that can be used in geographic information systems (GIS). Three sets of computational operations and the corresponding Galois lattices are presented. The choices made for spatial representation and reasoning with these lattices are discussed, together with a thorough study and comparison with a boolean lattice.

Lattices are interesting for spatial representation and reasoning since:

- They allow the classification of spatial relations and structures.
- They allow reasoning on relations and composition of relations: to reason on relations means to deduce new relations from already computed ones.
- They allow the definition of primitives to compute relations on raster or vector regions[2]: to compute –or to check– a relation R between two regions x and y means to test if $R(x, y)$ holds.

Lattices are hierarchical structures that are well adapted for knowledge representation [Mineau and Godin, 1995, Randell and Cohn, 1992, Wille, 1992]. They are used in object-based knowledge representation systems (ObKR systems) together with the inheritance mechanism and the classification process. The present work has been actually developed for an application in the domain of agricultural landscape analysis. One of the lattices has been implemented and evaluated in the framework of an ObKR system [Le Ber et al., 1999, Mangelinck, 1998].

Several new and original aspects in the management of spatial relations are noticeable in the present work. For the representation and reasoning purposes, the theory of Galois lattices has allowed to build different lattices providing different points of view on the topological relations, depending on the considered set of computational operations. Moreover, this work allows a qualitative as well as a quantitative approach to spatial reasoning, as proposed in [Galton, 1999, Haarslev et al., 1999]. Finally, it is

[1] The Galois lattice theory is also known as Formal Concept Analysis [Ganter and Wille, 1999, Wille, 1992].

[2] Raster regions are made of points while vector regions are defined with a polyline.

one of the unique works relying on the Galois lattice theory, that is usually associated with formal concept analysis [Wille, 1992, Ganter and Wille, 1999], or used for knowledge discovery purposes [Godin and Missaoui, 1994, Simon and Napoli, 1999].

This paper is structured as follows. The first part presents the Galois lattices of topological relations (design and examples). The second part is a study of the properties of the Galois lattices. The third part presents a comparison of the three Galois lattices together, with a boolean lattice and with other approaches for spatial representation and reasoning. The last part is a conclusion.

2 GALOIS LATTICES OF TOPOLOGICAL RELATIONS

The Galois lattices presented here are based on the implication relation between the eight base relations of **RCC-8** and a set of computational operations. The **RCC-8** relations are the following: $EQ(x,y)$: *"x is identical with y"*; $NTPP(x,y)$: *"x is a non tangential proper part of y"*; $TPP(x,y)$: *"x is a tangential proper part of y"*; $NTPP^{-1}(x,y)$: *"x non tangentially contains as a proper part y"*; $TPP^{-1}(x,y)$: *"x tangentially contains as a proper part y"*; $PO(x,y)$: *"x partially overlaps y"*; $EC(x,y)$: *"x is externally connected with y"*; $DC(x,y)$: *"x is disconnected from y"*. The set of these eight relations is denoted by $\mathcal{B}$. There exist several sets of computational operations that can be associated with the relations of $\mathcal{B}$ and that have been defined for different purposes. We first present three sets of computational operations from different authors, then the Galois lattice theory, and finally three Galois lattices of topological relations.

2.1 COMPUTATIONAL OPERATIONS

In order to deal with applications, such as geographic databases or image interpretation, one needs to compute relations, i.e. one needs operations for testing if a relation holds between two spatial regions. Such computational operations have been proposed in [Clementini et al., 1993, Egenhofer, 1989, Egenhofer and Sharma, 1993] for regular closed spatial regions: they are based on the notions of *interior* and *boundary* of a region. They are linked to formal models of the topological relations.

In [Egenhofer, 1989], computational operations are defined for the vector representation: a n-dimensional region x is characterized by two sets, the interior (of dimension n), denoted by x°, and the boundary (of dimension $n-1$), denoted by ∂x. Two regions are thus characterized by four sets whose intersections define four operations: $\partial x \cap \partial y$, $x^\circ \cap y^\circ$, $\partial x \cap y^\circ$ et $x^\circ \cap \partial y$. The relationship between the results of these operations and a set of topological relations is described in [Egenhofer, 1989]. These operations can be used in the same way to characterize **RCC-8** relations: for instance if the relation *"x partially overlaps y"*, $PO(x,y)$, holds between the two regions x and y, then the four operations applied to x and y give the following results: $\partial x \cap \partial y \neq \emptyset$, $x^\circ \cap y^\circ \neq \emptyset$, $\partial x \cap y^\circ \neq \emptyset$ and $x^\circ \cap \partial y \neq \emptyset$. Conversely, if the four operations give the previous results when applied to the two regions x and y, then the relation $PO(x,y)$ holds (Figure 1). The relationship between these operations and the base relations of **RCC-8** is described in Table 1. We call $CE\text{-}8$ the set of the eight conditions derived from these operations, according to their result (empty or not empty).

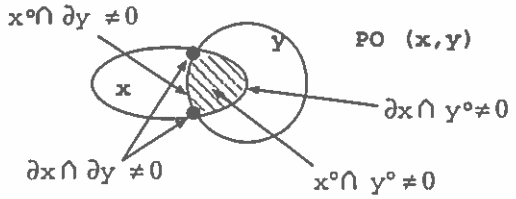


Figure 1: Relationship between the relation $PO(x,y)$ and the operations defined in [Egenhofer, 1989].

Table 1: Relationship between the **RCC-8** relations and the operations on interior and boundary sets proposed in [Egenhofer, 1989].

x,y	$\partial x \cap \partial y$	$x^\circ \cap y^\circ$	$\partial x \cap y^\circ$	$x^\circ \cap \partial y$
EQ	$\neq \emptyset$	$\neq \emptyset$	$\emptyset$	$\emptyset$
$NTPP$	$\emptyset$	$\neq \emptyset$	$\neq \emptyset$	$\emptyset$
TPP	$\neq \emptyset$	$\neq \emptyset$	$\neq \emptyset$	$\emptyset$
$NTPP^{-1}$	$\emptyset$	$\neq \emptyset$	$\emptyset$	$\neq \emptyset$
TPP^{-1}	$\neq \emptyset$	$\neq \emptyset$	$\emptyset$	$\neq \emptyset$
PO	$\neq \emptyset$	$\neq \emptyset$	$\neq \emptyset$	$\neq \emptyset$
EC	$\neq \emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
DC	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Another set of computational operations has been proposed in [Mangelinck, 1998] for regular closed raster regions. The four operations are: the intersection of the interior sets, $x^\circ \cap y^\circ$; the intersection of the boundary sets, $\partial x \cap \partial y$; the two differences of the interior sets, $x^\circ - y^\circ$ and $y^\circ - x^\circ$. From these operations are derived the eight following conditions:

- $(x^\circ - y^\circ = \emptyset)$, denoted by $P(x,y)$
- $(x^\circ - y^\circ \neq \emptyset)$, denoted by $Dx(x,y)$
- $(y^\circ - x^\circ = \emptyset)$, denoted by $P^{-1}(x,y)$
- $(y^\circ - x^\circ \neq \emptyset)$, denoted by $Dy(x,y)$
- $(x^\circ \cap y^\circ = \emptyset)$, denoted by $DR(x,y)$
- $(x^\circ \cap y^\circ \neq \emptyset)$, denoted by $O(x,y)$
- $(\partial x \cap \partial y = \emptyset)$, denoted by $NA(x,y)$
- $(\partial x \cap \partial y \neq \emptyset)$, denoted by $A(x,y)$

This set of conditions is called $CM\text{-}8$. The correspondence between each base relation and the conditions are described on Table 2: each relation is equivalent to a subset of conditions of $CM\text{-}8$.

Table 2: Relationship between the **RCC-8** relations and the computational operations proposed in [Mangelinck, 1998].

x,y	$x^\circ - y^\circ$	$y^\circ - x^\circ$	$x^\circ \cap y^\circ$	$\partial x \cap \partial y$
EQ	$\emptyset$	$\emptyset$	$\neq \emptyset$	$\neq \emptyset$
$NTPP$	$\emptyset$	$\neq \emptyset$	$\neq \emptyset$	$\emptyset$
TPP	$\emptyset$	$\neq \emptyset$	$\neq \emptyset$	$\neq \emptyset$
$NTPP^{-1}$	$\neq \emptyset$	$\emptyset$	$\neq \emptyset$	$\emptyset$
TPP^{-1}	$\neq \emptyset$	$\emptyset$	$\neq \emptyset$	$\neq \emptyset$
PO	$\neq \emptyset$	$\neq \emptyset$	$\neq \emptyset$	$\neq \emptyset$
EC	$\neq \emptyset$	$\neq \emptyset$	$\emptyset$	$\neq \emptyset$
DC	$\neq \emptyset$	$\neq \emptyset$	$\emptyset$	$\emptyset$

It has been also proposed to use the regions themselves and their interiors to compute the relations of **RCC-8** and **RCC-5** [Ligozat, 1999]. The operations are the intersection and the inclusion. For example, the relation $TPP(x,y)$ is defined with $\{x \subset y, x \not\subseteq y^\circ\}$. Moreover, there is a relationship between the set of relations $\mathcal{B}' = \{EQ, PP, PP^{-1}, PO, EC, DC\}$ and the set of conditions $CL\text{-}8 = \{x \subseteq y, x \not\subseteq y, y \subseteq x, y \not\subseteq x, x \cap y = \emptyset, x \cap y \neq \emptyset, x^\circ \cap y^\circ = \emptyset, x^\circ \cap y^\circ \neq \emptyset\}$ (Table 3).

Table 3: Relationship between the relations of $\mathcal{B}'$ and the conditions of $CL\text{-}8$. For the sake of simplicity, only four conditions are presented. A condition is true (t) or false (f).

x,y	$x \cap y = \emptyset$	$x^\circ \cap y^\circ = \emptyset$	$x \subseteq y$	$y \subseteq x$
EQ	f	f	t	t
PP	f	f	t	f
PP^{-1}	f	f	f	t
PO	f	f	f	f
EC	f	t	f	f
DC	t	t	f	f

2.2 THE DESIGN OF A GALOIS LATTICE

We present the design of a lattice $\mathcal{T}_{\mathcal{GM}}$ based on a Galois connection $\{f, g\}$ between the relation set $\mathcal{B}$ and the set of conditions $CM\text{-}8$ (details on the Galois lattice theory can be found in [Barbut and Monjardet, 1970, Davey and Priestley, 1990, Guénoche and Van Mechelen, 1993]). In the following, $\mathcal{D}$ is a set of regular spatial regions that are non empty, closed, and non necessarily internally connected (i.e. they may be made of several pieces), and $RCC\text{-}8$ is the powerset of $\mathcal{B}$.

Definition 1 *f is a binary relation mapping a relation r of $\mathcal{B}$ to a subset of $CM\text{-}8$ of all those conditions that hold for a pair (x,y) whenever the relation r holds for (x,y):*

$$f : \begin{array}{ccc} 2^{\mathcal{B}} & \mapsto & 2^{CM\text{-}8} \\ \{r\} & \mapsto & f(\{r\}) = \{c \in CM\text{-}8 \mid \\ & & \forall (x,y) \in \mathcal{D}^2 : r(x,y) \rightarrow c(x,y)\} \end{array}$$

Furthermore, f maps every subset R of $\mathcal{B}$ to the subset C of $CM\text{-}8$ of those conditions that are implied by *all* the relations of R: $f(R) = \cap_{r \in R} f(\{r\})$. The relation g is the dual of f:

Definition 2 *g is a binary relation mapping a condition c of $CM\text{-}8$ to the subset of $\mathcal{B}$ of all those relations that imply the condition c:*

$$g : \begin{array}{ccc} 2^{CM\text{-}8} & \mapsto & 2^{\mathcal{B}} = RCC\text{-}8 \\ \{c\} & \mapsto & g(\{r\}) = \{r \in \mathcal{B} \mid \\ & & \forall (x,y) \in \mathcal{D}^2 : r(x,y) \rightarrow c(x,y)\} \end{array}$$

Furthermore g maps every subset C of $CM\text{-}8$ to a subset R of $\mathcal{B}$ of those relations that imply *all* the conditions of C: $g(C) = \cap_{c \in C} g(\{c\})$. The Galois connection $\{f, g\}$ is expressed in Table 4.

Table 4: The table represents elements (r, c), with $r \in \mathcal{B}$ (row) and $c \in CM\text{-}8$ (column). An element $(r, c) = 1$ if $\forall (x,y) \in \mathcal{D}^2, r(x,y) \rightarrow c(x,y)$. Otherwise $(r, c) = 0$.

	P	Dx	P^{-1}	Dy	DR	O	NA	A
EQ	1	0	1	0	0	1	0	1
$NTPP$	1	0	0	1	0	1	1	0
TPP	1	0	0	1	0	1	0	1
$NTPP^{-1}$	0	1	1	0	0	1	1	0
TPP^{-1}	0	1	1	0	0	1	0	1
PO	0	1	0	1	0	1	0	1
EC	0	1	0	1	1	0	0	1
DC	0	1	0	1	1	0	1	0

The two functions f and g are used to define a closure operator h of $RCC\text{-}8$ and a closure operator h' of $2^{CM\text{-}8}$:

$$h : \begin{array}{ccc} RCC\text{-}8 & \mapsto & RCC\text{-}8 \\ h(R) & = & g \circ f(R) \end{array}$$

$$h' : \begin{array}{ccc} 2^{CM\text{-}8} & \mapsto & 2^{CM\text{-}8} \\ h'(C) & = & f \circ g(C) \end{array}$$

The Galois lattice $\mathcal{T}_{\mathcal{GM}}$ is built on the basis of the two closure operators h and h'. Its elements are pairs $(C, R) \in 2^{CM\text{-}8} \times RCC\text{-}8$, where $h(R) = R$ and $C = f(R) = h'(C)$. Such a pair can also be considered as a formal concept derived from the implication

relation between $\mathcal{B}$ and $CM\text{-}8$, C being the intension and R the extension of this concept. $\mathcal{T}_{\mathcal{GM}}$ is defined as follows:

Definition 3 *The Galois lattice $\mathcal{T}_{\mathcal{GM}}$ based on the implication between the relations of $\mathcal{B}$ and the conditions of $CM\text{-}8$ is the structure $\langle \mathcal{E}_{\mathcal{G}}, \sqsubseteq, \frown, \smile, (CM\text{-}8, \emptyset), (\emptyset, \mathcal{B})\rangle$ where:*

- *$\mathcal{E}_{\mathcal{G}}$ is the set of all pairs (C, R) where R is a subset of $\mathcal{B}$ closed for h, C is a subset of $CM\text{-}8$ closed for h', $f(R) = C$ and $g(C) = R$.*
- *The ordering $\sqsubseteq$ between two elements (C_1, R_1) and (C_2, R_2) is defined as follows:*

$$(C_1, R_1) \sqsubseteq (C_2, R_2) \leftrightarrow \begin{cases} C_2 \subseteq C_1 \\ R_1 \subseteq R_2 \end{cases}$$

  where $\subseteq$ is the set-inclusion.
- *The greatest lower bound (denoted by $\frown$) of two elements is defined as follows:*

$$(C_1, R_1) \frown (C_2, R_2) = (h'(C_1 \cup C_2), R_1 \cap R_2)$$

  where $\cup$ and $\cap$ are the set-union and the set-intersection.
- *The least upper bound (denoted by $\smile$) of two elements is defined as follows:*

$$(C_1, R_1) \smile (C_2, R_2) = (C_1 \cap C_2, h(R_1 \cup R_2))$$

- *The bottom element is $(CM\text{-}8, \emptyset)$,*
- *The top element is $(\emptyset, \mathcal{B})$.*

In a more concrete way, the Galois lattice is built by searching for the maximal rectangles in the table of f (see Table 4) or by searching for the subsets of $\mathcal{B}$ that are closed for h. In particular, all the singletons $\{r\}$ of $RCC\text{-}8$ are closed for h, *i.e.* $h(\{r\}) = \{r\}$.

In order to exploit this lattice, we now define the π_r function that maps an element (C, R) of $\mathcal{T}_{\mathcal{GM}}$ to the disjunction of the relations of R (denoted $\pi_r(C, R)$):

$$\forall (x, y) \in \mathcal{D}^2, \ \pi_r(C, R)(x, y) =_{def} \bigvee_{r \in R} r(x, y)$$

Similarly we introduce the π_c function, that is concerned with the conditions of $CM\text{-}8$. The π_c function maps an element (C, R) of $\mathcal{T}_{\mathcal{GM}}$ to the conjunction of the C elements:

$$\forall (x, y) \in \mathcal{D}^2, \ \pi_c(C, R)(x, y) =_{def} \bigwedge_{c \in C} c(x, y)$$

We show below that the two formulas resulting from the application of π_c and π_r are equivalent. This equivalence is used to name the elements of $\mathcal{T}_{\mathcal{GM}}$, either as a disjunction of relations or as a conjunction of conditions. The π_c and π_r functions are also used to make explicit the link between the lattice ordering and the implication relation on the topological relations. Actually, if an element (C_1, R_1) (e.g. `EQ`, see Figure 2 below) is smaller than another element $E_2 = (C_2, R_2)$ (e.g. `TP`), the relation $\pi_r(C_1, R_1)$ implies the relation $\pi_r(C_2, R_2)$ and the condition $\pi_c(C_1, R_1)$ implies the condition $\pi_c(C_2, R_2)$ [Le Ber et al., 2001].

2.3 THREE GALOIS LATTICES OF TOPOLOGICAL RELATIONS

The first lattice $\mathcal{T}_{\mathcal{GM}}$ is based on the set of conditions $CM\text{-}8$ proposed in [Mangelinck, 1998] and is described on Figure 2. Its elements are denoted by a name or an icon: a name represents a conjunction of conditions (e.g. `OetA` denotes the element $(\{O, A\}, R)$, where `et` stands for "and") or a disjunction of relations (e.g. `PP` denotes the element $(C, \{TPP, NTPP\})$, since $PP = TPP \vee NTTP$)[3]. The icons at the bottom of the lattice represent the eight base relations of **RCC-8** (e.g. `EQ` $= (\{O, A, P, P^{-1}\}, \{EQ\})$.

Another Galois lattice based on the computational operations proposed in [Egenhofer, 1989] is described hereafter. Firstly we give a name to the eight associated conditions. Four of these conditions are identical to those of $CM\text{-}8$ (A, NA, DR, O). The four others are:

$\partial x \cap y^\circ = \emptyset$	denoted by	NFO
$\partial x \cap y^\circ \neq \emptyset$	denoted by	FO
$x^\circ \cap \partial y = \emptyset$	denoted by	NOF
$x^\circ \cap \partial y \neq \emptyset$	denoted by	OF

The Galois connection f between the sets $\mathcal{B}$ and $CE\text{-}8$ is made explicit in Table 5. The lattice is named $\mathcal{T}_{\mathcal{GE}}$, its elements are pairs (C, R), where C is a subset of $CE\text{-}8$ and R is a subset of $\mathcal{B}$. It is displayed in Figure 3.

Table 5: The Galois connection between the $\mathcal{B}$ relations and the $CE\text{-}8$ conditions.

	DR/O	NA/A	NFO/FO	NOF/OF
EQ	0 / 1	0 / 1	1 / 0	1 / 0
$NTPP$	0 / 1	1 / 0	0 / 1	1 / 0
TPP	0 / 1	0 / 1	0 / 1	1 / 0
$NTPP^{-1}$	0 / 1	1 / 0	1 / 0	0 / 1
TPP^{-1}	0 / 1	0 / 1	1 / 0	0 / 1
PO	0 / 1	0 / 1	0 / 1	0 / 1
EC	1 / 0	0 / 1	1 / 0	1 / 0
DC	1 / 0	1 / 0	1 / 0	1 / 0

A third lattice has been built thanks to the computational operations proposed in [Ligozat, 1999]. The eight associated conditions are the following:

[3] In the following A denotes the condition and `A` denotes the lattice element where the condition is represented (respectively EC and `EC` to denote the topological relation and the element of the lattice).

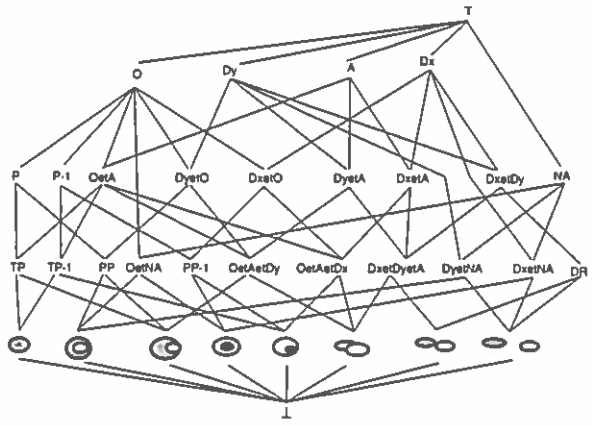


Figure 2: The Galois lattice based on CM-8 and $\mathcal{B}$. The pairs (C, R) are denoted by the relation, or by the condition, to which they are associated by the relations π_r or π_c [Mangelinck, 1998].

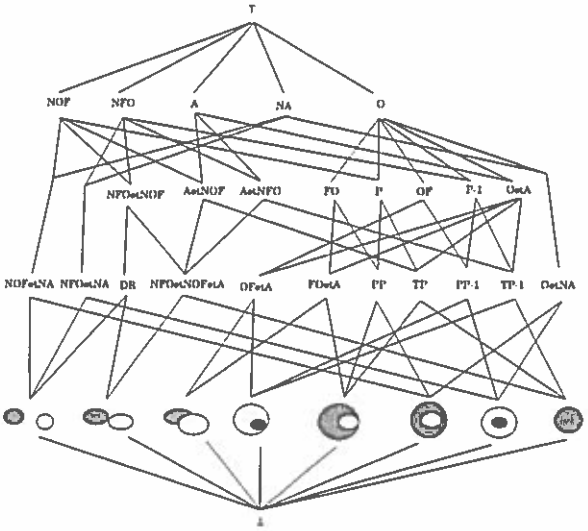


Figure 3: The Galois lattice based on CE-8 and $\mathcal{B}$.

$x \cap y = \emptyset$	corresponding to	$DC(x, y)$
$x \cap y \neq \emptyset$	corresponding to	$C(x, y)$
$x^\circ \cap y^\circ = \emptyset$	corresponding to	$DR(x, y)$
$x^\circ \cap y^\circ \neq \emptyset$	corresponding to	$O(x, y)$
$x \subseteq y$	corresponding to	$P(x, y)$
$x \not\subseteq y$	corresponding to	$Dx(x, y)$
$y \subseteq x$	corresponding to	$P^{-1}(x, y)$
$y \not\subseteq x$	corresponding to	$Dy(x, y)$

Two conditions are similar to those of CE-8 (DR and O) and six are similar to those of CM-8. Only two conditions, C and DC, make CL-8 distinct from the other sets of conditions. The lattice $\mathcal{T}_{\mathcal{GL}}$ is built thanks to the Galois connection between CL-8 and the relation set $\mathcal{B}' = \{DC, EC, PO, PP^{-1}, PP, EQ\}$. It is displayed in Figure 4.

3 SOME PROPERTIES OF GALOIS LATTICES

In this section, we present general properties of the lattice $\mathcal{T}_{\mathcal{GM}}$. Actually, these properties are not particular to $\mathcal{T}_{\mathcal{GM}}$ and they can be read in the framework of the two other Galois lattices. The proofs of this properties are given in [Le Ber et al., 2001].

3.1 THE CHARACTERISTICS OF THE ELEMENTS OF $\mathcal{T}_{\mathcal{GM}}$

The following properties characterize the elements of the $\mathcal{T}_{\mathcal{GM}}$ lattice:

Property 1 *If a pair* $\mathtt{E} = (C, R)$ *is an element of the* $\mathcal{T}_{\mathcal{GM}}$ *lattice, then the relation* $\pi_r(\mathtt{E}) = \pi_r(C, R)$

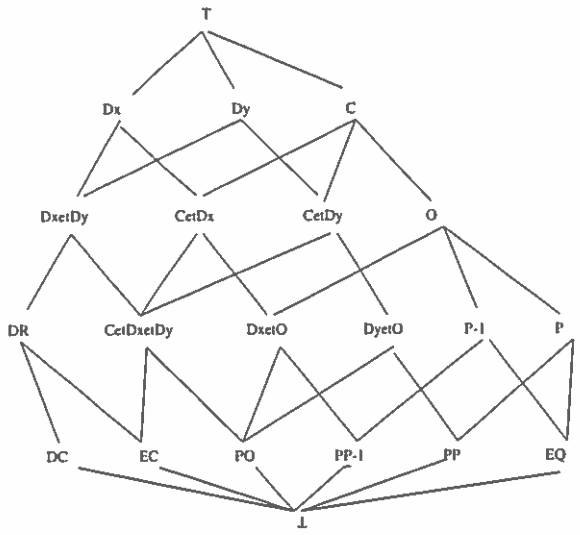


Figure 4: The Galois lattice built on CL-8 and $\mathcal{B}$'.

is equivalent to the condition $\pi_c(\mathtt{E}) = \pi_c(C, R)$*:*

$$(C, R) \in \mathcal{E}_\mathcal{G} \rightarrow$$
$$(\forall (x, y) \in \mathcal{D}^2, \bigvee_{r \in R} r(x, y) \leftrightarrow \bigwedge_{c \in C} c(x, y))$$

This property shows that the relation $\pi_r(\mathtt{E})$ associated with an element of $\mathcal{T}_{\mathcal{GM}}$ is logically equivalent to the condition $\pi_c(\mathtt{E})$ associated with the same element. Thus, checking a relation on the image requires to look for the element of $\mathcal{T}_{\mathcal{GM}}$ that represents this relation and to check the associated conditions. The second property that characterizes the elements of the $\mathcal{T}_{\mathcal{GM}}$ lattice is as follows:

Property 2 *Let* R *be a subset of* $\mathcal{B}$ *and* C *a subset of* CM*-8. If the relation formed by the disjunction of the elements of* R *is equivalent to the condition formed by the conjunction of the elements of* C*, then the pair* $(\bar{C}, R)$*, where* $\bar{C} = h'(C)$*, is an element of* $\mathcal{T}_{\mathcal{GM}}$*:*

$$(\forall (x, y) \in \mathcal{D}^2, \bigvee_{r \in R} r(x, y) \leftrightarrow \bigwedge_{c \in C} c(x, y)) \rightarrow (\bar{C}, R) \in \mathcal{T}_{\mathcal{GM}}$$

The next property characterizes the elements of $\mathcal{B}$. It can be deduced from the property 2 or by examining Table 4:

Property 3 *For each element* r *of* $\mathcal{B}$*,* $(f(\{r\}), \{r\})$ *is an element of* $\mathcal{T}_{\mathcal{GM}}$*.*

Property 3 shows that the base relations of $\mathcal{B}$ are represented as individuals in the $\mathcal{T}_{\mathcal{GM}}$ lattice, situated at the bottom of the lattice, as displayed in Figure 2. The next property characterizes the conditions of CM-8 and their representation in $\mathcal{T}_{\mathcal{GM}}$:

Property 4 *Each condition* c *of* CM*-8 can be associated to a distinct element of* $\mathcal{T}_{\mathcal{GM}}$*,* $\mathtt{E_c} = (\bar{C}, R)$*, where* $R = g(\{c\})$*,* $\bar{C} = h'(\{c\})$ *and:*

$$\forall (x, y) \in \mathcal{D}^2, c(x, y) \leftrightarrow \bigvee_{r \in R} r(x, y) \leftrightarrow \bigwedge_{c_i \in \bar{C}} c_i(x, y)$$

This last property can directly be checked on the conditions of CM-8. In particular:

$$\begin{array}{ll} g(\{P\}) = \{EQ, NTPP, TPP\} & \\ h'(\{P\}) = (P, O) & \text{and} \\ P(x, y) \leftrightarrow (P \wedge O)(x, y) & \\ g(\{P^{-1}\}) = \{EQ, NTPP^{-1}, TPP^{-1}\} & \\ h'(\{P^{-1}\}) = (P^{-1}, O) & \text{and} \\ P^{-1}(x, y) \leftrightarrow (P^{-1} \wedge O)(x, y) & \\ g(\{DR\}) = \{DC, EC\} & \\ h'(\{DR\}) = (DR, Dx, Dy) & \text{and} \\ DR(x, y) \leftrightarrow (DR \wedge Dx \wedge Dy)(x, y) & \end{array}$$

The other singletons of $2^{CM\text{-}8}$ are closed for h' : $h'(\{c\}) = \{c\}$. Thus, all conditions of CM-8 are represented as individuals in the $\mathcal{T}_{\mathcal{GM}}$ lattice. They are represented respectively by the elements P, $\mathtt{P^{-1}}$, DR, O, Dy, A, Dx and NA (see Figure 2).

3.2 CHARACTERIZING THE *glb* ($\frown$) AND THE *lub* ($\smile$)

Property 5 *An element* E *of* $\mathcal{T}_{\mathcal{GM}}$ *is the glb of two elements* $\mathtt{E_1}$ *and* $\mathtt{E_2}$ *if and only if* $\pi_r(\mathtt{E})$ *is equivalent to the conjunction of the two relations* $\pi_r(\mathtt{E_1})$ *and* $\pi_r(\mathtt{E_2})$*:*

$$(\mathtt{E} = \mathtt{E_1} \frown \mathtt{E_2}) \leftrightarrow (\forall (x, y) \in \mathcal{D}^2,$$
$$\pi_r(\mathtt{E})(\mathtt{x}, \mathtt{y}) \leftrightarrow \pi_r(\mathtt{E_1})(\mathtt{x}, \mathtt{y}) \wedge \pi_r(\mathtt{E_2})(\mathtt{x}, \mathtt{y}))$$

Furthermore, each element E of $\mathcal{T}_{\mathcal{GM}}$ is characterized by the equivalence between the disjunction $\pi_r(\mathtt{E})$ and the conjunction $\pi_c(\mathtt{E})$. Thus:

Property 6 *Let* E, E_1, E_2 *be three elements of the* $\mathcal{T}_{\mathcal{GM}}$ *lattice:*

$$(E = E_1 \frown E_2) \leftrightarrow (\forall (x,y) \in \mathcal{D}^2,$$
$$\pi_c(E)(x,y) \leftrightarrow \pi_c(E_1)(x,y) \wedge \pi_c(E_2)(x,y))$$

Property 7 *For all pairs of lattice elements* (E_1, E_2), *the lub* $E' = E_1 \smile E_2$ *is such that the disjunction of* $\pi_r(E_1)$ *and* $\pi_r(E_2)$ *implies the relation* $\pi_r(E')$*:*

$$(E' = E_1 \smile E_2) \rightarrow (\forall (x,y) \in \mathcal{D}^2,$$
$$\pi_r(E')(x,y) \leftarrow \pi_r(E_1)(x,y) \vee \pi_r(E_2)(x,y))$$

This property is easy to prove since the relation set R' of the element $E' = E_1 \smile E_2$ is the closure of the union of the two relation sets R_1, R_2 of the elements E_1 and E_2 (cf. the definition of the lattice). The reciprocal property of Property 7 is not true in the $\mathcal{T}_{\mathcal{GM}}$ lattice because an element of $\mathcal{T}_{\mathcal{GM}}$ can be the *lub* of several elements: E' can be both the *lub* of the pair E_1, E_2 and the *lub* of the pair E_3, E_4. Then:

$$E' = E_1 \smile E_2 : \pi_r(E_1) \vee \pi_r(E_2) \rightarrow \pi_r(E')$$
$$E' = E_3 \smile E_4 : \pi_r(E_2) \vee \pi_r(E_4) \rightarrow \pi_r(E')$$
$$\text{and } \pi_r(E') \rightarrow \pi_r(E_1) \vee \pi_r(E_2) \vee \pi_r(E_3) \vee \ldots$$

For instance, the *lub* of the elements TP and TP^{-1} of $\mathcal{T}_{\mathcal{GM}}$ is the element OetA which is also the *lub* of the elements OetAetDx and OetAetDy (see Figure 2). This property has to be linked to a property of closed sets: the *lub* of two closed sets is generally not a closed set, whereas the *glb* is a closed set [Barbut and Monjardet, 1970, Davey and Priestley, 1990].

3.3 THE MINIMAL SET OF CONDITIONS FOR CHECKING A RELATION

The relations expressed in the $\mathcal{T}_{\mathcal{GM}}$ lattice are associated to a set C of conditions which are sufficient to check this relation. But all conditions of C are not necessary, and some of these conditions could be eliminated to minimize computation. Let us take an example: $EQ(x,y)$, *"x is identical with y"*, is associated to the set of conditions $C = \{A, O, P, P^{-1}\}$ (see Table 4). But, the checking of $EQ(x,y)$ only relies on the checking of the conditions $P(x,y)$ (*"x is a part of y"*) and $P^{-1}(x,y)$ (*"x contains y"*) since the conjunction of these two conditions imply the two other conditions $A(x,y)$ (*"x shares a boundary with y"*) and $O(x,y)$ (*"x overlaps y"*). This implication is due to the properties of the image regions (non-emptiness, closure) as previously mentioned. In $\mathcal{T}_{\mathcal{GM}}$, EQ is represented by the element EQ $= (\{A, O, P, P^{-1}\}, \{EQ\})$. According to Property 1:

$$\forall (x,y) \in \mathcal{D}^2,\ EQ(x,y) \leftrightarrow (A \wedge O \wedge P \wedge P^{-1})(x,y)$$

Furthermore, P and P^{-1} are represented by the elements P and P^{-1}. The *glb* of P and P^{-1} is EQ (see Figure 2). According to Properties 6 and 4:

$$\forall (x,y) \in \mathcal{D}^2 :$$
$$\pi_c(EQ)(x,y) \leftrightarrow \pi_c(P)(x,y) \wedge \pi_c(P^{-1})(x,y)$$
$$(A \wedge O \wedge P \wedge P^{-1})(x,y) \leftrightarrow (P \wedge P^{-1})(x,y)$$
$$EQ(x,y) \leftrightarrow (P \wedge P^{-1})(x,y)$$

Finally, the properties of the operator $\frown$ can be used to find out the minimal set of conditions that are necessary to check the EQ relation. This minimal set can directly be read on the lattice.

The minimal sets associated to the other relations of $\mathcal{B}$ can be found accordingly:

$$\begin{aligned}
\forall (x,y) \in \mathcal{D}^2 :& \\
NTPP(x,y) &\leftrightarrow (P \wedge NA)(x,y) \\
TPP(x,y) &\leftrightarrow (P \wedge A \wedge Dy)(x,y) \\
NTPP^{-1}(x,y) &\leftrightarrow (P^{-1} \wedge NA)(x,y) \\
TPP^{-1}(x,y) &\leftrightarrow (P^{-1} \wedge A \wedge Dx)(x,y) \\
PO(x,y) &\leftrightarrow (Dx \wedge Dy \wedge O)(x,y) \\
EC(x,y) &\leftrightarrow (A \wedge DR)(x,y) \\
DC(x,y) &\leftrightarrow (NA \wedge DR)(x,y)
\end{aligned}$$

It should be noticed that these conditions can be associated with appropriate algorithmic operations for testing topological relations on quantitative data. An example is given for raster images in [Mangelinck, 1998, Le Ber et al., 1999]. Such an approach is also described in [Bennett et al., 1998] for vector data.

4 A COMPARISON OF LATTICES

4.1 THE GALOIS LATTICES VERSUS THE BOOLEAN LATTICE $\mathcal{T}_{\mathcal{P}}$

We first compare the Galois lattices with the lattice $\mathcal{T}_{\mathcal{P}} = \langle RCC\text{-}8, \subseteq, \cap, \cup, \emptyset, \mathcal{B} \rangle$ that has been used for theorem proving and to infer rules of composition between relations [Randell and Cohn, 1992]. The lattice $\mathcal{T}_{\mathcal{P}}$ is a boolean lattice (i.e. distributive and complemented) and its cardinality is $2^8 = 256$.

From an implementation point of view, the Galois lattices are more easy to manage than the $\mathcal{T}_{\mathcal{P}}$ lattice. For instance, an object-based knowledge representation system must be reasonably sized in terms of memory space. The $\mathcal{T}_{\mathcal{P}}$ lattice contains 256 elements, which means that the designer has to implement and to manage 256 classes of relations. The $\mathcal{T}_{\mathcal{GM}}$ lattice on the contrary contains only 34 elements (see Figure 2). Thus, it can be worth to choose the $\mathcal{T}_{\mathcal{GM}}$ lattice rather than the $\mathcal{T}_{\mathcal{P}}$ lattice for an application.

Another major advantage of the Galois lattices is that they include the representation of the conditions used to check the relations on spatial regions, and the representation of the implication relations holding between

the conditions and the relations. This is not the case in the $\mathcal{T}_{\mathcal{P}}$ lattice: the conditions are only implicitly represented, as they are equivalent to disjunctions of $\mathcal{B}$ relations. Thus, the implication relations between the topological relations and the conditions do not appear in the $\mathcal{T}_{\mathcal{P}}$ lattice, making this lattice not very useful for checking relations on the image. On the contrary, the links between relations and conditions are singled out in the $\mathcal{T}_{\mathcal{GM}}$ and $\mathcal{T}_{\mathcal{GE}}$ lattices, since they are built from the Galois connection between $\mathcal{B}$ and CM-8 or CE-8 (respectively).

The last point is about reasoning in the lattices: the reasoning is complete in $\mathcal{T}_{\mathcal{P}}$ whereas it is not in Galois lattices. One of the reasons is that some compositions of the base relations are lacking in the Galois lattices. In this case, the $\mathcal{T}_{\mathcal{P}}$ lattice has to be preferred: it includes all the disjunctions of $\mathcal{B}$ elements and thus contains all the relations of the composition table of $\mathcal{B}$. By contrast, the $\mathcal{T}_{\mathcal{GM}}$ lattice lacks three compositions. In order to combine the advantages of $\mathcal{T}_{\mathcal{GM}}$ (computation) and the advantages of $\mathcal{T}_{\mathcal{P}}$ (inference), it is possible to extend the $\mathcal{T}_{\mathcal{GM}}$ lattice with the lacking compositions, but the resulting lattice is no longer a Galois lattice [Le Ber et al., 2001].

The incompleteness of the reasoning in Galois lattices is also due to the fact that these lattices are not complemented and that the *lub* of two elements is not equivalent to the disjunction of the corresponding relations (see paragraph 3.2). For instance, if the relation $(TP \vee TP^{-1})(x,y)$ holds, it is concluded in $\mathcal{T}_{\mathcal{GM}}$ that $(TPP \vee TPP^{-1} \vee EQ \vee PO)(x,y)$ holds (DetA is the *lub* of TP and TP^{-1}), whereas it is concluded in $\mathcal{T}_{\mathcal{P}}$ that $(TPP \vee TPP^{-1} \vee EQ)(x,y)$ holds. The last conclusion cannot be found in $\mathcal{T}_{\mathcal{GM}}$ since the corresponding element does not exist.

4.2 $\mathcal{T}_{\mathcal{GM}}$ VERSUS $\mathcal{T}_{\mathcal{GE}}$ AND $\mathcal{T}_{\mathcal{GL}}$

Since they are all Galois lattices, the $\mathcal{T}_{\mathcal{GM}}$, $\mathcal{T}_{\mathcal{GE}}$ and $\mathcal{T}_{\mathcal{GL}}$ lattices have basically the same properties. The differences rely on the condition sets (CM-8, CE-8, CL-8) and the relation sets ($\mathcal{B}$, $\mathcal{B}$') they are based on. In addition, the sets of elements of the three lattices are different.

Let us first examine the conditions of CE-8. Five of them are expressed within elements that are immediate descendants of $\top$, the top element of the lattice (see Figure 3). These elements are A, O, NA, NFO, NOF, whose set of conditions, $C = \{c\}$, is closed for h':

$$\forall c \in \{A,\ O,\ NA,\ NOF,\ NFO\},\ h'(\{c\}) = \{c\}$$

The other conditions behave as follows:

$$h'(\{OF\}) = \{OF, O\} \text{ and}$$
$$OF(x,y) \leftrightarrow (OF \wedge O)(x,y)$$
$$h'(\{FO\}) = \{FO, O\} \text{ and}$$
$$FO(x,y) \leftrightarrow (FO \wedge O)(x,y)$$
$$h'(\{DR\}) = \{DR, NOF, NFO\} \text{ and}$$
$$DR(x,y) \leftrightarrow (DR \wedge NOF \wedge NFO)(x,y)$$

In $\mathcal{T}_{\mathcal{GL}}$, only three conditions, C, Dx, Dy, are expressed within elements that are immediate descendants of $\top$. The other conditions behave this way:

$$h'(\{O\}) = \{O, C\} \text{ and}$$
$$O(x,y) \leftrightarrow (O \wedge C)(x,y)$$
$$h'(\{P^{-1}\}) = \{P^{-1}, C, O\} \text{ and}$$
$$P^{-1}(x,y) \leftrightarrow (P^{-1} \wedge C \wedge O)(x,y)$$
$$h'(\{P\}) = \{P, C, O\} \text{ and}$$
$$P(x,y) \leftrightarrow (P \wedge C \wedge O)(x,y)$$
$$h'(\{DR\}) = \{DR, Dx, Dy\} \text{ and}$$
$$DR(x,y) \leftrightarrow (DR \wedge Dx \wedge Dy)(x,y)$$
$$h'(\{DC\}) = \{DC, DR, Dx, Dy\} \text{ and}$$
$$DC(x,y) \leftrightarrow (DC \wedge DR \wedge Dx \wedge Dy)(x,y)$$

The CL-8 set of conditions seems finally to be more 'redundant' or less efficient than the two others. The condition $x \cap y = \emptyset$ in particular can only be used to check the base relation $DC(x,y)$ and nothing else.

We now compare the number of conditions used to check the relations in $\mathcal{T}_{\mathcal{GE}}$ and $\mathcal{T}_{\mathcal{GM}}$, since these two lattices are based on the same relation set $\mathcal{B}$. This comparison is done for the base relations in Table 6. The first column gives the names of the corresponding elements in the two lattices. The two following columns give the sets of conditions C_E (in $\mathcal{T}_{\mathcal{GE}}$) and C_M (in $\mathcal{T}_{\mathcal{GM}}$). The two last columns give the number (n_E, n_M) of conditions used to check the relations. The number of conditions used to check all the relations expressed in the two lattices is given in [Le Ber et al., 2001].

Checking a relation does not require the same number of conditions in the two lattices: for instance, checking the relation EQ requires two conditions in $\mathcal{T}_{\mathcal{GM}}$, $\{P, P^{-1}\}$ (see section 3.3), whereas it needs three conditions in $\mathcal{T}_{\mathcal{GE}}$, $\{O, NFO, NOF\}$. Conversely, checking the relation PO relies on three conditions in $\mathcal{T}_{\mathcal{GM}}$, $\{Dx, Dy, A\}$, whereas it needs only the two conditions $\{FO, OF\}$ in $\mathcal{T}_{\mathcal{GE}}$.

To summarize, the two lattices $\mathcal{T}_{\mathcal{GE}}$ and $\mathcal{T}_{\mathcal{GM}}$ use the same average number of conditions to check the relations. Thus, the choice between these two lattices is not a matter of optimization (i.e. minimization of the number of tests) but rather a matter of adequacy with the needs of the current application data: in one hand, it can be preferable to compute the intersection of a boundary set and an interior set ($\partial x \cap y^{\circ}$, $x^{\circ} \cap \partial y$), in the other hand, it can be preferable to compute the difference of two interior sets ($x^{\circ} - y^{\circ}$, $y^{\circ} - x^{\circ}$), as discussed for raster data in [Le Ber et al., 2001].

4.3 DISCUSSION

Lattices have been used for theorem proving and relations inferring in the framework of **RCC-8**

Table 6: The number of conditions used to check the base relations in $\mathcal{T}_{\mathcal{GE}}$ and $\mathcal{T}_{\mathcal{GM}}$.

E_E / E_M	C_E	C_M	n_E	n_M
EQ	$\{NFO, NOF, O, A\}$	$\{P, P^{-1}, O, A\}$	3	2
NTPP	$\{FO, O, NA, NOF\}$	$\{P, O, NA, Dy\}$	2	2
TPP	$\{FO, O, A, NOF\}$	$\{P, O, A, Dy\}$	3	3
NTPP^{-1}	$\{OF, O, NA, NFO\}$	$\{P^{-1}, O, NA, Dx\}$	2	2
TPP^{-1}	$\{OF, O, A, NFO\}$	$\{P^{-1}, O, A, Dx\}$	3	3
PO	$\{FO, OF, O, A\}$	$\{Dx, Dy, O, A\}$	2	3
EC	$\{DR, NOF, NFO, A\}$	$\{DR, Dx, Dy, A\}$	2	2
DC	$\{DR, NOF, NFO, NA\}$	$\{DR, Dx, Dy, NA\}$	2	2

[Randell and Cohn, 1992]. Moreover, they are well adapted for classification based reasoning. The Galois lattices in particular are well adapted for classification and computation of topological relations. Their main drawback is that reasoning is not complete. This drawback is in balance with the low number of elements in the Galois lattices, making them more manageable than the boolean lattice $\mathcal{T}_{\mathcal{P}}$. Finally, Galois lattices can be used with benefits for applications involving relation computation and classification.

An important reasoning problem in the Region Connection Calculus, denoted by RSAT, is deciding the consistency of a set of constraints of the form xRy, where R is a disjunction of the base relations and x, y are spatial regions. This problem is NP-hard in general, but there exist subsets of *RCC-8* where it is tractable [Renz and Nebel, 1999, Renz, 1999]. Three maximal tractable subsets of *RCC-8* containing all base relations are defined in [Renz, 1999]. According to these results, we can conclude that RSAT($\mathcal{T}_{\mathcal{GM}}$) and RSAT($\mathcal{T}_{\mathcal{GE}}$) are NP-hard since the two lattices contain the relations $\{NTPP, NTPP^{-1}\}$ (element OetNA) and $\{DC, NTPP, NTPP^{-1}\}$ (NA). All other relations represented in $\mathcal{T}_{\mathcal{GM}}$ and $\mathcal{T}_{\mathcal{GE}}$ are member of the maximal tractable subset $\mathcal{H}_8$ defined in [Renz and Nebel, 1999, Renz, 1999].

Finally, let us mention works having objectives similar to ours. The works in [Haarslev et al., 1998] and [Haarslev et al., 1999] hold on qualitative spatial representation for managing map databases and spatial query processing. Spatial reasoning relies on two main operations in terminological reasoning, namely consistency checking and classification. In [Möller and Wessel, 1999], spatiotemporal default reasoning is introduced: a specific query completion problem is studied and default knowledge is used for completing and making more precise queries.

5 CONCLUSION

In this paper, we have presented three Galois lattices that represent topological relations for spatial reasoning. The Galois lattice theory provides a powerful framework for classifying topological relations, and for designing a connection between computational conditions –linked with real-world images and numerical data– and topological relations of **RCC-8** –used for qualitative spatial reasoning. This connection is of main importance in the domain of spatial reasoning, and geographical information systems. The Galois lattices presented in this paper can be extended to take into account the whole composition set of spatial relations. They can also be used according to different points of view on topological relations and computational conditions, e.g. vector vs. raster regions.

Finally, Galois lattices are not only theoretical tools, and they provide practical means for implementing knowledge-based systems. Actually, in our framework, an implementation based on a frame system has been carried out, that gives satisfying results in the field of landscape analysis [Mangelinck, 1998, Le Ber et al., 1999].

References

[Barbut and Monjardet, 1970] Barbut, M. and Monjardet, B. (1970). *Ordre et classification – Algèbre et combinatoire*. Hachette, Paris.

[Bennett et al., 1998] Bennett, B., Isli, A., and Cohn, A. G. (1998). A system handling RCC-8 queries on 2D regions representable in the closure algebra of half-planes. In *Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA-AIE)*, LNCS. Springer-Verlag.

[Clementini et al., 1993] Clementini, E., Di Felice, P., and van Oosterom, P. (1993). A small set of formal topological relationships for end-user interaction. In Abel, D. J. and Ooi, B. C., editors, *Advances in Spatial Databases: Proceedings of the Third International Symposium, SSD'93*, LNCS 692, pages 277–295, Singapore. Springer Verlag.

[Cohn et al., 1997] Cohn, A. G., Bennett, B., Gooday, J., and Gotts, N. M. (1997). Representing and reasoning with qualitative spatial relations about regions. In [Stock, 1997], pages 97–134.

[Davey and Priestley, 1990] Davey, B. and Priestley, H. (1990). *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, UK.

[Egenhofer, 1989] Egenhofer, M. J. (1989). A Formal Definition of Binary Topological Relationships. In Litwin, W. and Schek, H., editors, *Foundations of data organization and algorithms, proceedings of the 3rd international conference, FODO 1989*, LNCS 367, pages 457–472. Springer Verlag.

[Egenhofer and Sharma, 1993] Egenhofer, M. J. and Sharma, J. (1993). Topological Relations Between Regions in $\Re^2$ and Z^2. In Abel, D. J. and Ooi, B. C., editors, *Advances in spatial databases: Proceedings of the Third International Symposium, SSD'93*, LNCS 692, pages 316–336. Springer Verlag.

[Freksa and Mark, 1999] Freksa, C. and Mark, D. M., editors (1999). *Spatial Information Theory, Cognitive and Computational Foundations of Geographic Information Science*, LNCS 1661. Springer. International Conference COSIT'99.

[Galton, 1999] Galton, A. (1999). The mereology of discrete space. In [Freksa and Mark, 1999], pages 251–266.

[Ganter and Wille, 1999] Ganter, B. and Wille, R. (1999). *Formal Concept Analysis*. Springer, Berlin.

[Godin and Missaoui, 1994] Godin, R. and Missaoui, R. (1994). An incremental concept formation approach for learning from databases. *Theoretical Computer Science*, 133(2):387–419.

[Guénoche and Van Mechelen, 1993] Guénoche, A. and Van Mechelen, I. (1993). Galois Approach to the Induction of Concepts. In Van Mechelen, I., Hampton, J., Michalski, R., and Theuns, P., editors, *Categories and Concepts. Theoretical Views and Inductive Data Analysis*, pages 287–308. Academic Press, London.

[Haarslev et al., 1998] Haarslev, V., Lutz, C., and Möller, R. (1998). Foundations of Spatioterminological Reasoning with Description Logics. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning KR'98*, pages 112–123. Morgan Kaufmann.

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

[Le Ber et al., 1999] Le Ber, F., Mangelinck, L., and Napoli, A. (1999). Représentation de relations et classification de structures spatiales. *Revue d'Intelligence Artificielle*, 13(2):441–467.

[Le Ber et al., 2001] Le Ber, F., Mangelinck, L., and Napoli, A. (2001). Design and comparison of lattices of topological relations for spatial representation and reasoning. INRIA Research Report RR-4321.

[Ligozat, 1999] Ligozat, G. (1999). Simple Models for Simple Calculi. In [Freksa and Mark, 1999], pages 173–188.

[Mangelinck, 1998] Mangelinck, L. (1998). Représentation et classification de structures spatiales. Application à la reconnaissance de paysages agricoles. Thèse de doctorat, Université Henri Poincaré Nancy 1.

[Mineau and Godin, 1995] Mineau, G. and Godin, R. (1995). Automatic Structuring of Knowledge Bases by Conceptual Clustering. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):824–829.

[Möller and Wessel, 1999] Möller, R. and Wessel, M. (1999). Terminological Default Reasoning about spatial information: A First Step. In [Freksa and Mark, 1999], pages 189–204.

[Randell and Cohn, 1992] Randell, D. A. and Cohn, A. G. (1992). Exploiting Lattices in a Theory of Space and Time. *Computers Math. Applic.*, 23(6-9):459–476.

[Randell et al., 1992] Randell, D. A., Cui, Z., and Cohn, A. G. (1992). A Spatial Logic based on Regions and Connection. In *3rd International Conference on Knowledge Representation and Reasoning*, pages 165–176. Morgan Kaufmann.

[Renz, 1999] Renz, J. (1999). Maximal tractable fragment of the region connection calculus: a complete analysis. In *Proceedings of the 16th* IJCAI, pages 448–454.

[Renz and Nebel, 1999] Renz, J. and 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.

[Simon and Napoli, 1999] Simon, A. and Napoli, A. (1999). Building viewpoints in an object-based representation system for knowledge discovery in databases. In *Proceedings of the First International Conference on Information Reuse and Integration (IRI'99)*, pages 104–108. The International Society for Computers and Their Applications, ISCA.

[Stock, 1997] Stock, O., editor (1997). *Spatial and Temporal Reasoning*. Kluwer Academic Publishers.

[Wille, 1992] Wille, R. (1992). Concept lattices and conceptual knowledge systems. *Computers & Mathematics With Applications*, 23(6–9):493–515.

Complexity Issues I

Complexity Results for Explanations in the Structural-Model Approach

Thomas Eiter
Institut für Informationssysteme,
Technische Universität Wien
Favoritenstraße 9-11, 1040 Wien, Austria
eiter@kr.tuwien.ac.at

Thomas Lukasiewicz*
Dipartimento di Informatica e Sistemistica,
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
lukasiewicz@dis.uniroma1.it

Abstract

We analyze the computational complexity of Halpern and Pearl's (causal) explanations in the structural-model approach, which are based on their notions of weak and actual causality. In particular, we give a precise picture of the complexity of deciding explanations, α-partial explanations, and partial explanations, and of computing the explanatory power of partial explanations. Moreover, we analyze the complexity of deciding whether an explanation or an α-partial explanation over certain variables exists. We also analyze the complexity of deciding explanations and partial explanations in the case of succinctly represented context sets, and the complexity of deciding explanations in the general case of situations. All complexity results are derived for the general case, as well as for the restriction to the case of binary causal models, in which all endogenous variables may take only two values. To our knowledge, no complexity results for explanations in the structural-model approach have been derived so far. Our results give insight into the computational structure of Halpern and Pearl's explanations, and pave the way for efficient algorithms and implementations.

1 INTRODUCTION

The automatic generation of explanations for humans is of crucial importance in areas like planning, diagnosis, natural language processing, and probabilistic inference. Notions of explanations have been studied quite extensively in the literature, see especially [21, 14, 36] for philosophical work, and [25, 38, 22] for work in AI that is related to Bayesian networks. A critical examination of such approaches from the viewpoint of explanations in probabilistic systems is given in [4].

In a recent paper [18, 20], Halpern and Pearl introduced an elegant definition of causal explanation in the structural-model approach, which is based on their notions of weak and actual causality [18, 19]. They showed that this notion of causal explanation models well many problematic examples in the literature. Note that Halpern and Pearl's causal explanation is very different from the concept of causal explanation in [28, 29, 15].

The following example from [18, 19, 20] illustrates the structural-model approach. See especially [1, 13, 31, 32, 17] for more details on structural causal models.

Example 1.1 *(arsonists)* Suppose two arsonists lit matches in different parts of a dry forest, and both cause trees to start burning. Assume now either match by itself suffices to burn down the whole forest. We may model such a scenario in the structural-model framework as follows. We assume two binary background variables U_1 and U_2, which determine the motivation and the state of mind of the two arsonists, where U_i is 1 iff arsonist i intends to start a fire. We then have three binary variables A_1, A_2, and B, which describe the observable situation, where A_i is 1 iff arsonist i drops the match, and B is 1 iff the whole forest burns down. The causal dependencies between these variables are expressed by functions, which say that the value of A_i is given by the value of U_i, and that B is 1 iff either A_1 or A_2 is 1. These dependencies can be graphically represented as in Fig. 1.

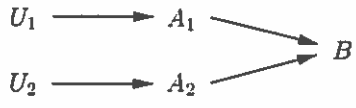


Figure 1: Causal Graph

*Alternate address: Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, 1040 Wien, Austria. E-mail: lukasiewicz@kr.tuwien.ac.at.

While the semantic aspects of explanations in the structural-model approach have been thoroughly studied in [18, 20], an analysis of their computational properties is missing so far. In this paper, we fill this gap. Our main contributions are summarized as follows:

- We analyze the computational complexity of Halpern and Pearl's explanations in the structural-model approach [18, 20]. In particular, we draw a precise picture of the complexity of deciding explanations, α-partial explanations, and partial explanations, and of computing the explanatory power of partial explanations.
- We also analyze the complexity of the natural problem of deciding whether an explanation or an α-partial explanation over certain variables exists.
- We show that deciding explanations and partial explanations has a higher complexity in the case of succinctly represented context sets. Generalizing from contexts to situations, in contrast, does not increase the complexity of deciding explanations.
- We also show that all analyzed problems have a lower complexity in the binary case.

Note that detailed proofs of all results are given in the extended paper [10].

2 PRELIMINARIES

We assume a finite set of *random variables*. Each variable X_i may take on *values* from a finite *domain* $D(X_i)$. A *value* for a set of variables $X = \{X_1, \ldots, X_n\}$ is a mapping $x\colon X \to D(X_1) \cup \cdots \cup D(X_n)$ such that $x(X_i) \in D(X_i)$ (for $X = \emptyset$, the unique value is the empty mapping $\emptyset$). The *domain* of X, denoted $D(X)$, is the set of all values for X. For $Y \subseteq X$ and $x \in D(X)$, denote by $x|Y$ the restriction of x to Y. For sets of variables X, Y and values $x \in D(X), y \in D(Y)$, denote by xy the union of x and y. We often identify singletons $\{X_i\}$ with X_i, and their values x with $x(X_i)$.

2.1 CAUSAL MODELS

A *causal model* $M = (U, V, F)$ consists of two disjoint finite sets U and V of *exogenous* and *endogenous* variables, respectively, and a set $F = \{F_X \mid X \in V\}$ of functions $F_X\colon D(PA_X) \to D(X)$ that assign a value of X to each value of the *parents* $PA_X \subseteq U \cup V - \{X\}$ of X.

We focus here on the principal class [18] of *recursive* causal models $M = (U, V, F)$ in which a total ordering $\prec$ on V exists such that $Y \in PA_X$ implies $Y \prec X$, for all $X, Y \in V$. In such models, every assignment to the exogenous variables $U = u$ determines a unique value y for every set of endogenous variables $Y \subseteq V$, denoted $Y_M(u)$ (or simply $Y(u)$). In the sequel, M is reserved for denoting a recursive causal model. For any causal model $M = (U, V, F)$, set of variables $X \subseteq V$, and $x \in D(X)$, the causal model $M_x = (U, V, F_x)$, where $F_x = \{F_Y \mid Y \in V - X\} \cup \{F_{X'} = x(X') \mid X' \in X\}$, is a *submodel* of M. For $Y \subseteq V$, we abbreviate $Y_{M_x}(u)$ by $Y_x(u)$. We say $M = (U, V, F)$ is *binary* iff $|D(X)| = 2$ for all $X \in V$.

Example 2.1 *(arsonists continued)* $M = (U, V, F)$ for Example 1.1 is given by $U = \{U_1, U_2\}$, $V = \{A_1, A_2, B\}$, and $F = \{F_{A_1}, F_{A_2}, F_B\}$, where $F_{A_1} = U_1$, $F_{A_2} = U_2$, and $F_B = 1$ iff $A_1 = 1$ or $A_2 = 1$ (Fig. 1 shows the parent relationship between the variables).

As for computation, we assume that in $M = (U, V, F)$, every function $F_X\colon D(PA_X) \to D(X)$, $X \in V$, is computable in polynomial time. The following is immediate.

Proposition 2.1 *For all $X, Y \subseteq V$ and $x \in D(X)$, the values $Y(u)$ and $Y_x(u)$, given $u \in D(U)$, are computable in polynomial time.*

2.2 CAUSALITY

We now recall weak causes from [18, 19]. A *primitive event* is an expression of the form $Y = y$, where Y is a variable and y is a value for Y. The set of *events* is the closure of the set of primitive events under the Boolean operations $\neg$ and $\wedge$. The *truth* of an event ϕ in $M = (U, V, F)$ under $u \in D(U)$, denoted $(M, u) \models \phi$, is inductively defined by:

- $(M, u) \models Y = y$ iff $Y_M(u) = y$,
- $(M, u) \models \neg\phi$ iff $(M, u) \models \phi$ does not hold,
- $(M, u) \models \phi \wedge \psi$ iff $(M, u) \models \phi$ and $(M, u) \models \psi$.

We write $\phi(u)$ to abbreviate $(M, u) \models \phi$. For $X \subseteq V$ and $x \in D(X)$, we write $\phi_x(u)$ to abbreviate $(M_x, u) \models \phi$. For $X = \{X_1, \ldots, X_k\} \subseteq V$ with $k \geq 1$ and $x_i \in D(X_i)$, we use $X = x_1 \cdots x_k$ to abbreviate $X_1 = x_1 \wedge \ldots \wedge X_k = x_k$. The following is immediate.

Proposition 2.2 *Let $X \subseteq V$ and $x \in D(X)$. Given $u \in D(U)$ and an event ϕ, deciding whether $\phi(u)$ and $\phi_x(u)$ (given x) hold can be done in polynomial time.*

Let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$ and $x \in D(X)$, and let ϕ be an event. Then, $X = x$ is a *weak cause* of ϕ under u iff the following conditions hold:

AC1. $X(u) = x$ and $\phi(u)$.

AC2. Some set of variables $W \subseteq V - X$ and some values $\overline{x} \in D(X)$, $w \in D(W)$ exist with:

(a) $\neg\phi_{\overline{x}w}(u)$, and
(b) $\phi_{xw\hat{z}}(u)$ for all $\hat{Z} \subseteq V - (X \cup W)$ and $\hat{z} = \hat{Z}(u)$.

Example 2.2 *(arsonists continued)* Consider the context $u_{1,1} = (1,1)$ in which both arsonists intend to start a fire. Then, $A_1 = 1$, $A_2 = 1$, and $A_1 = 1 \wedge A_2 = 1$ are weak causes of $B = 1$. Moreover, $A_1 = 1$ (resp., $A_2 = 1$) is the only weak cause of $B = 1$ under the context $u_{1,0} = (1,0)$ (resp., $u_{0,1} = (0,1)$) in which only arsonist 1 (resp., 2) intends to start a fire.

The following lemma characterizes irrelevant variables in weak causes.

Lemma 2.3 *Let $M = (U, V, F)$. Let $X \subseteq V$ and $x \in D(X)$, let ϕ be an event, and let $u \in D(U)$. Let $X_0 \in V$ such that in the causal network for M, it holds that X_0 is not a predecessor of any variable in ϕ, and $X_0(u) = x(X_0)$. Let $X' = X - \{X_0\}$ and $x' = x|X'$. Then, $X = x$ is a weak cause of ϕ under u iff $X' = x'$ is a weak cause of ϕ under u.*

We recall a result from [11, 12], which shows that deciding weak cause is complete for Σ_2^{P} (resp., NP) in the general (resp., binary) case. Note that this result holds also when the domain $D(X) = \{1, \ldots, n_X\}$ of each variable $X \in U \cup V$ is specified by $n_X \geq 1$.

Theorem 2.4 (see [11, 12]) *Given $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, $u \in D(U)$, and an event ϕ, deciding whether $X = x$ is a weak cause of ϕ under u is complete for Σ_2^{P} (resp., NP) in the general (resp., binary) case.*

2.3 EXPLANATION

We now recall the concept of explanation from [18, 20]. Let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$ and $x \in D(X)$, ϕ be an event, and $\mathcal{C} \subseteq D(U)$ be a set of contexts. Then, $X = x$ is an *explanation* of ϕ relative to $\mathcal{C}$ iff the following conditions hold:

EX1. $\phi(u)$ for each context $u \in \mathcal{C}$.

EX2. $X = x$ is a weak cause of ϕ under every $u \in \mathcal{C}$ such that $X(u) = x$.

EX3. X is minimal. That is, for every $X' \subset X$, some $u \in \mathcal{C}$ exists such that $X'(u) = x|X'$ and $X' = x|X'$ is not a weak cause of ϕ under u.

EX4. $X(u) = x$ for some $u \in \mathcal{C}$, and $X(u') \neq x$ for some $u' \in \mathcal{C}$.

Example 2.3 *(arsonists continued)* Consider the set of contexts $\mathcal{C} = \{u_{1,1}, u_{1,0}, u_{0,1}\}$. Then, both $A_1 = 1$ and $A_2 = 1$ are explanations of $B = 1$ relative to $\mathcal{C}$, while $A_1 = 1 \wedge A_2 = 1$ is not, as here, the minimality condition EX3 is violated.

2.4 PARTIAL EXPLANATION AND EXPLANATORY POWER

We finally recall the notions of partial and α-partial explanations and of explanatory power [18, 20]. Let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$ and $x \in D(X)$, let ϕ be an event, let $\mathcal{C} \subseteq D(U)$ such that $\phi(u)$ for all $u \in \mathcal{C}$. We use the expression $\mathcal{C}^{\phi}_{X=x}$ to denote the unique largest subset $\mathcal{C}'$ of $\mathcal{C}$ such that $X = x$ is an explanation of ϕ relative to $\mathcal{C}'$; it is easy to see that if such a set $\mathcal{C}'$ exists, then $\mathcal{C}^{\phi}_{X=x}$ is defined. Let P be a probability function on $\mathcal{C}$, and define

$$P(\mathcal{C}^{\phi}_{X=x} \mid X = x) = \sum_{u \in \mathcal{C}^{\phi}_{X=x},\, X(u)=x} P(u) \;/\; \sum_{u \in \mathcal{C},\, X(u)=x} P(u).$$

Then, $X = x$ is called an *α-partial explanation* of ϕ relative to $(\mathcal{C}, P)$ iff $\mathcal{C}^{\phi}_{X=x}$ is defined and $P(\mathcal{C}^{\phi}_{X=x} \mid X = x) \geq \alpha$. We say $X = x$ is a *partial explanation* of ϕ relative to $(\mathcal{C}, P)$ iff $X = x$ is an *α-partial explanation* of ϕ relative to $(\mathcal{C}, P)$ for some $\alpha > 0$; furthermore, $P(\mathcal{C}^{\phi}_{X=x} \mid X = x)$ is called its *explanatory power* (or *goodness*).

Example 2.4 *(arsonists continued)* Let $\mathcal{C} = \{u_{1,1}, u_{1,0}, u_{0,1}\}$, and let P be the uniform distribution over $\mathcal{C}$. Then, both $A_1 = 1$ and $A_2 = 1$ are 1-partial explanations of $B = 1$. That is, both $A_1 = 1$ and $A_2 = 1$ are partial explanations of $B = 1$ with explanatory power 1.

2.5 COMPLEXITY CLASSES

The complexity classes that we encounter are shown in Fig. 2. They are well-known classes from the Polynomial Hierarchy (PH), or derived from them. We recall that $\mathrm{NP} = \Sigma_1^{\mathrm{P}}$, co-$\mathrm{NP} = \Pi_1^{\mathrm{P}}$, $\Sigma_{k+1}^{\mathrm{P}} = \mathrm{NP}^{\Sigma_k^{\mathrm{P}}}$, and $\Pi_k^{\mathrm{P}} =$ co-Σ_k^{P}, $k \geq 1$, are classes in PH. The class $\mathrm{D}_k^{\mathrm{P}} = \{L \times L' \mid L \in \Sigma_k^{\mathrm{P}}, L' \in \Pi_k^{\mathrm{P}}\}$, $k \geq 1$, is the "conjunction" of Σ_k^{P} and Π_k^{P}; in particular, $\mathrm{D}_1^{\mathrm{P}}$ is the familiar class D^{P}. The class $\mathrm{P}_{\|}^{\Sigma_k^{\mathrm{P}}}$, $k \geq 1$, contains the decision problems which can be solved in polynomial time with parallel calls to a Σ_k^{P} oracle; $\mathrm{FP}_{\|}^{\Sigma_k^{\mathrm{P}}}$ is the analog for function computations. Note that $\mathrm{P}_{\|}^{\mathrm{NP}} = \mathrm{P}_{\|}^{\Sigma_1^{\mathrm{P}}}$ and $\mathrm{FP}_{\|}^{\mathrm{NP}} = \mathrm{FP}_{\|}^{\Sigma_1^{\mathrm{P}}}$. For further background on the complexity classes, see e.g. [23, 24, 30, 41].

3 OVERVIEW OF RESULTS

In this section, we give an overview on the complexity results that we derive, and discuss possible implications.

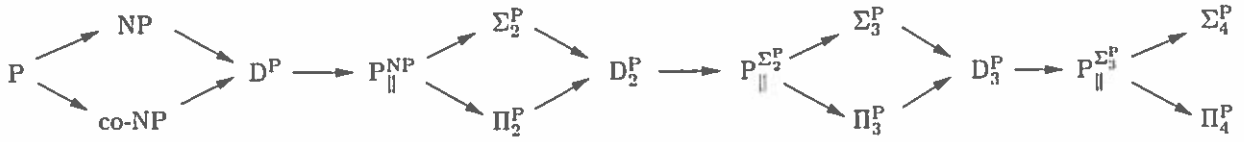


Figure 2: Containment between Complexity Classes

3.1 PROBLEM STATEMENTS

In our analysis, we focus on the following problems, which are major tasks in explanation-based causal reasoning:

Explanation: Given $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, an event ϕ, and a set of contexts $\mathcal{C} \subseteq D(U)$, decide whether $X = x$ is an explanation of ϕ relative to $\mathcal{C}$.

Explanation Existence: Given $M = (U, V, F)$, $X \subseteq V$, an event ϕ, and a set of contexts $\mathcal{C} \subseteq D(U)$, decide whether some $X' \subseteq X$ and $x' \in D(X')$ exist such that $X' = x'$ is an explanation of ϕ relative to $\mathcal{C}$.

α-Partial Explanation: Given $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, an event ϕ, a set of contexts $\mathcal{C} \subseteq D(U)$ such that $\phi(u)$ for all $u \in \mathcal{C}$, a probability function P on $\mathcal{C}$, and $\alpha \geq 0$, decide whether $X = x$ is an α-partial explanation of ϕ relative to $(\mathcal{C}, P)$.

α-Partial Explanation Existence: Given $M = (U, V, F)$, $X \subseteq V$, an event ϕ, a set of contexts $\mathcal{C} \subseteq D(U)$ such that $\phi(u)$ for all $u \in \mathcal{C}$, a probability function P on $\mathcal{C}$, and $\alpha \geq 0$, decide whether some $X' \subseteq X$ and $x' \in D(X')$ exist such that $X' = x'$ is an α-partial explanation of ϕ relative to $(\mathcal{C}, P)$.

Partial Explanation: Given $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, an event ϕ, a set of contexts $\mathcal{C} \subseteq D(U)$ such that $\phi(u)$ for all $u \in \mathcal{C}$, a probability function P on $\mathcal{C}$, decide whether $X = x$ is a partial explanation of ϕ relative to $(\mathcal{C}, P)$.

Explanatory Power: Given $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, an event ϕ, $\mathcal{C} \subseteq D(U)$, and a probability function P on $\mathcal{C}$, where (i) $\phi(u)$ for all $u \in \mathcal{C}$, and (i) $X = x$ is a partial explanation of ϕ relative to $(\mathcal{C}, P)$, compute the explanatory power of $X = x$.

In all problems, the probability function P is assumed to be polynomially computable.

The first problem, Explanation, is the recognition of an explanation. It emerges directly from the definition of explanation in Section 2.3 and captures its intrinsic complexity. The problem Explanation Existence is associated with the important task of finding an explanation for an event ϕ. Similar as in other frameworks for explanations (e.g. [27, 37]), the set X focuses attention to a subset of the variables, in terms of which the explanation must be formed. Finding explanations is certainly the central task of a causal-reasoning system built for applications in practice, and thus this problem deserves special attention. The problems α-Partial / Partial Explanation and α-Partial Explanation Existence can be viewed as relaxations of Explanation and Explanation Existence, respectively, in a probabilistic context. Explanatory Power is the problem of computing the "goodness" of a partial explanation $X = x$, given by the coverage of the cases where $X = x$ is true in the contexts $\mathcal{C}$. This information can be used to rank partial explanations and single out "best" ones.

Table 1: Complexity of Explanations

Problem	general case	binary case
Explanation	D_2^P-complete	D^P-complete
Explanation Existence	Σ_3^P-complete	Σ_2^P-complete
α-Partial Explanation	$P_{\|}^{\Sigma_2^P}$-complete	$P_{\|}^{NP}$-complete
α-Partial Explanation Existence	Σ_3^P-complete	Σ_2^P-complete
Partial Explanation	$P_{\|}^{\Sigma_2^P}$-complete	$P_{\|}^{NP}$-complete
Explanatory Power	$FP_{\|}^{\Sigma_2^P}$-complete	$FP_{\|}^{NP}$-complete

Table 2: Complexity of Explanations: Succinct Contexts

Problem	general case	binary case
Explanation	Π_4^P-complete	Π_3^P-complete
Partial Explanation	Π_4^P-complete	Π_3^P-complete

Table 3: Complexity of Explanations: Situations

Problem	general case	binary case
Explanation	D_2^P-complete	D^P-complete

3.2 MAIN RESULTS

Our main complexity results are compactly summarized in Tables 1–3. Besides the general case, they include results for binary causal models, and also address succinct context representation (see Table 2) and a generalization from

contexts to situations in [20] (see Table 3).

All results in Tables 1–3 show completeness under standard polynomial-time transformations [24, 30], and thus sharply characterize the complexity of the problems. From the results in Table 1, it appears that finding explanations and α-partial explanations is at the third level of PH. Thus, explanations are harder to compute than weak causes, which lie at the second level of PH [11]. On the other hand, recognizing explanations and α-partial explanations is only mildly harder than recognizing weak causes, which is Σ_2^{P}-complete. The reason is that by the latter, condition EX2 amounts to a conjunction of a linear number of problems in Σ_2^{P}, and EX3 to the negation of such a problem; EX1 and EX4 are easily checked. Thus, by usual techniques, the explanation check can be reduced to a conjunction of problems in Σ_2^{P} and Π_2^{P}. In the case of an α-partial explanation, we need to know the context $C^{\phi}_{X=x}$; by exploiting a basic characterization result (Lemma 4.3), it can be computed efficiently with parallel calls to a Σ_2^{P} oracle. Once $C^{\phi}_{X=x}$ is known, we need to check whether $X = x$ is an explanation relative to it, the rest is easy. Thus, the complexity of this problem, as well as of Explanatory Power, lies here in the computation of $C^{\phi}_{X=x}$. The Σ_3^{P} upper bound for Explanation Existence and α-Partial Explanation Existence is then straightforward by a standard guess and check argument.

The Σ_3^{P}-hardness of Explanation Existence stems from a subtlety in the definition of explanation. From satisfaction of EX1, EX2, and EX4 for $X = x$ we can *not* conclude that some $X' = x'$ contained in $X = x$ exists which will satisfy EX1-EX4; if we minimize $X = x$ so as to satisfy EX3, the resulting $X' = x'$ may violate EX4. It is this interplay of the conditions which makes this problem difficult, and the proofs of the hardness results nontrivial. The Σ_3^{P}-hardness of α-Partial Explanation Existence is inherited from the hardness of Explanation Existence.

3.3 SUCCINCT CONTEXTS

Table 2 shows results for some of the problems in a setting where contexts are succinctly represented. In fact, Table 1 assumes that the set of contexts $\mathcal{C}$ is enumerated in the input. However, $\mathcal{C}$ may contain exponentially many contexts; a descriptive representation can be much more compact and desirable in practice. In the succinct representation setting, we thus assume that $\mathcal{C}$ is given by a tractable membership function $\chi_{\mathcal{C}}(u)$. That is, on input of $u \in D(U)$, function $\chi_{\mathcal{C}}(u)$ reports in polynomial time whether $u \in \mathcal{C}$ holds. This includes, e.g., descriptions of $\mathcal{C}$ in terms of propositional formulas β over U such that the models of β describe the contexts in $\mathcal{C}$. It turns out that succinct representation increases the complexity of Explanation and α-Partial Explanation drastically. Intuitively, in this case checking a property for all contexts in $\mathcal{C}$ becomes much harder, since there seems no better way than guessing the "right" context witnessing or disproving the property. The complexity increase by two levels in PH stems from the fact that condition EX3 involves two nested checks of properties for all contexts in $\mathcal{C}$. This dominates the complexity of EX1, EX2, and EX4 and leads to Π_4^{P} complexity. For α-partial explanations, we have similar effects. Worse, we need to calculate sums of probabilities over succinctly represented context sets. This leads us outside PH: It requires to solve problems which are at least as hard deciding whether a given propositional CNF β has $\geq k$ models, where k is in the input. This problem is, as generally believed, not in PH. We refrain from a detailed analysis of computing α-partial explanations here. For Partial Explanation, we obtain the entry shown in the table. A complexity increase for Explanation Existence under succinct context sets to Σ_5^{P} is plausible, though we have not analyzed it yet; note that already the Π_4^{P}-hardness proof for Explanation is rather involved.

3.4 SITUATIONS

Table 3 shows results for the generalization of explanations from contexts to situations discussed in [20]. Without going into the details here, in this scenario the epistemic state consists of a set $\mathcal{S}$ of pairs (M, u) called *situations*, where M is a causal model and u is a context, rather than a set of contexts. Informally, a situation (M, u) encodes some causal knowledge in M and some known facts in u. General explanations are then defined as pairs $(\psi, X = x)$ where ψ is an arbitrary causal formula that restricts the causal models to be considered, and $X = x$ is a conjunction of primitive events. The definition is similar to the one of explanations, and is too involved to be presented here (see [20]); it covers basic explanations as a special case. Interestingly, general explanations are not more difficult to recognize than basic explanations.

3.5 RESTRICTED CASES

This concludes our exposition of the complexity results in the general case. Tables 1-3 also show results for the restriction to *binary causal models*, where each endogenous variable may take only two values. In this case, the complexity of all considered problems drops by one level in PH; this parallels the drop of the complexity of weak causes from Σ_2^{P} to NP in the binary case [11]. The membership parts can be derived analogously as in the general case, and the hardness parts by slight adaptations of the constructions in the proofs, where certain subcomponents for weak cause testing are modularly replaced.

Some of our hardness results remain valid under further restrictions, such as a boundedness condition on the causal model [11, 12]. In particular, all hardness results from Tables 1-3 hold for primitive events ϕ; thus, complex events

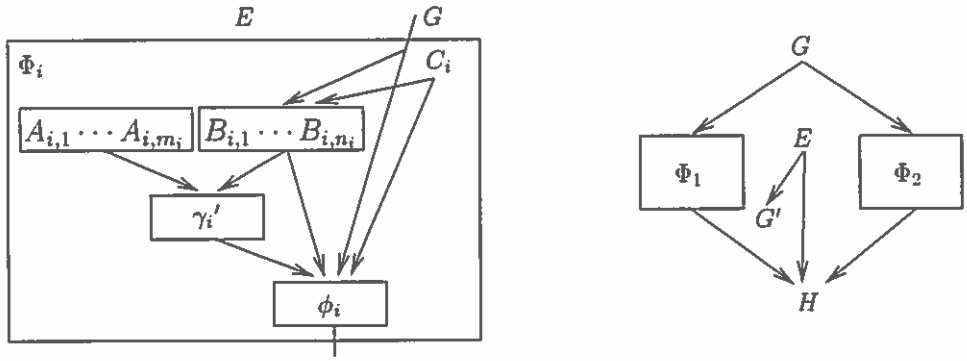


Figure 3: Schematic Construction for Evaluating two QBFs Φ_1 and Φ_2

are not a source of complexity. To avoid a proliferation of results, we do not further consider restrictions here.

3.6 IMPLICATIONS FOR COMPUTATION

For "efficient" algorithms that generate explanations or "best" α-partial explanations, we can draw the following conclusions. Both must solve an inherent Σ_3^P-hard problem, i.e., a problem at the third level of PH; such problems are rather hard to solve. Informally, the problem is "triple NP-hard:" even if we could use a subroutine for solving Σ_2^P-complete problems for free, the problem would be intractable (NP-hard). Similarly, Σ_2^P-complete problems are intractable even if we could use a subroutine for solving NP-complete problems for free. Thus, computationally speaking, generating explanations is rather difficult. In particular, a simple NP-style backtracking strategy that explores, similar as a simple Davis-Putnam style SAT-solver, a polynomial-depth search tree is infeasible. By similar arguments, polynomial-time reductions to a SAT-solver or a computational logic system which can handle problems with complexity up to Σ_2^P, such as DLV [9] are infeasible.

On the other hand, an explanation can be computed using a nested backtracking procedure (modeling nested subroutine calls), or using flat backtracking calling a subroutine for Σ_2^P tasks (e.g., calls to DLV). A further possible perspective are translations to QBF-solvers, which proved valuable in other applications [33]. We can compute an α-partial explanation similarly. Computing a best one amounts to an optimization problem, which can be solved by binary search over the range [0,1] of α, and thus in polynomial time with a Σ_3^P oracle. A substantially faster algorithm seems unlikely to exist.

4 DERIVATION OF RESULTS

We now sketch how some of our complexity results can be formally derived. More detailed proofs are given in Appendix A. Detailed proofs of all results are given in [10]. Many of these proofs are technically quite involved.

4.1 EXPLANATION

Theorem 4.1 *Explanation is D_2^P-complete.*

Proof (sketch). As for membership in D_2^P, recall that $X = x$ is an explanation of ϕ relative to $\mathcal{C}$ iff EX1–EX4 hold. Deciding in EX1 whether $\phi(u)$ for every $u \in \mathcal{C}$ and in EX4 whether $X(u) = x$ and $X(u') \neq x$ for some $u, u' \in \mathcal{C}$ is polynomial. In EX2, the set $\mathcal{C}'$ of all $u \in \mathcal{C}$ such that $X(u) = x$ is polynomially computable. By Theorem 2.4 and as Σ_2^P is closed under polynomially many conjunctions, deciding whether $X = x$ is a weak cause of ϕ under every $u \in \mathcal{C}'$ is in Σ_2^P. In EX3, guessing some $X' \subset X$ and checking that $X' = x|X'$ is a weak cause of ϕ under every $u \in \mathcal{C}$ such that $X'(u) = x|X'$ is in Σ_2^P. Thus, deciding EX3 is in Π_2^P. In summary, deciding whether $X = x$ is an explanation of ϕ relative to $\mathcal{C}$ is in D_2^P.

Hardness for D_2^P is shown by a reduction from deciding, given a pair (Φ_1, Φ_2) of QBFs $\Phi_i = \exists A_i \forall B_i \gamma_i$ with $i \in \{1,2\}$, where each γ_i is a propositional formula on the variables $A_i = \{A_{i,1}, \ldots, A_{i,m_i}\}$ and $B_i = \{B_{i,1}, \ldots, B_{i,n_i}\}$, whether Φ_1 is valid and Φ_2 is not valid. We build $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, $\mathcal{C} \subseteq D(U)$, and ϕ as required such that $X = x$ is an explanation of ϕ relative to $\mathcal{C}$ iff Φ_1 is valid and Φ_2 is not valid. Roughly, the main idea behind this construction is as follows. We construct $M_1 = (U, V_1, F_1)$ and $M_2 = (U, V_2, F_2)$ and two events ϕ_1 and ϕ_2 such that (i) $V_1 \cap V_2 = \{G\}$, and (ii) for every $u \in D(U)$, it holds that $G = 0$ is a weak cause of ϕ_i under u in M_i iff Φ_i is valid (see Fig. 3, left side). The causal model M is the union of M_1 and M_2, enlarged by additional endogenous variables (see Fig. 3, right side). We then construct ϕ and $u_1, u_2 \in D(U)$ such that ϕ is under u_1 and u_2 equivalent to ϕ_1 and ϕ_2, respectively. Finally, the construction is

such that $G=0 \wedge G'=0$ is an explanation of ϕ relative to $C=\{u_1,u_2\}$ in M, iff (a) $G=0$ is a weak cause of ϕ_1 under u_1 in M_1, and (b) $G=0$ is not a weak cause of ϕ_2 under u_2 in M_2, where (a) (resp., (b)) is encoded in EX2 (resp., EX3). That is, $G=0 \wedge G'=0$ is an explanation of ϕ relative to C in M, iff Φ_1 is valid and Φ_2 is not valid.

Theorem 4.2 *Explanation Existence is Σ_3^P-complete.*

Proof (sketch). We guess some $X' \subseteq X$ and $x' \in D(X')$, and verify that $X'=x'$ is an explanation of ϕ relative to C. By Theorem 4.1, this can be done in polynomial time with two calls to a Σ_2^P-oracle. Thus, the problem is in Σ_3^P.

Σ_3^P-hardness is shown by a reduction from deciding whether a given QBF $\Phi=\exists B \forall C \exists D \gamma$ is valid, where γ is a propositional formula on the variables $B \cup C \cup D$. We construct $M=(U,V,F)$, $X \subseteq V$, $C \subseteq D(U)$, and ϕ such that Φ is valid iff some $X' \subseteq X$ and $x' \in D(X')$ exist such that $X'=x'$ is an explanation of ϕ relative to C. Roughly, the main idea is to encode the quantor "$\exists B$" in guessing some $X' \subseteq X$, and "$\forall C \exists D \gamma$" in checking the complement of a weak cause in EX3. Note that the construction is technically involved. □

4.2 PARTIAL EXPLANATION AND EXPLANATORY POWER

We now focus on the complexity of deciding partial and α-partial explanations. The following lemma gives a useful characterization of the set $C^{\phi}_{X=x}$, which is used below.

Lemma 4.3 *Let $M=(U,V,F)$ be a causal model. Let $X \subseteq V$ and $x \in D(X)$, and let ϕ be an event. Let $C \subseteq D(U)$ such that $\phi(u)$ for all $u \in C$. Then, $C^{\phi}_{X=x}$ is the set of all $u \in C$ such that either (i) $X(u) \neq x$, or (ii) $X(u)=x$ and $X=x$ is a weak cause of ϕ under u.*

Theorem 4.4 *α-Partial Explanation is $\mathrm{P}_{\parallel}^{\Sigma_2^P}$-complete.*

Proof (sketch). We first prove membership in $\mathrm{P}_{\parallel}^{\Sigma_2^P}$. Recall that $X=x$ is an α-partial explanation of ϕ relative to (C,P) iff (a) $X=x$ is an explanation of ϕ relative to $C^{\phi}_{X=x}$, and (b) $P(C^{\phi}_{X=x} \mid X=x) \geq \alpha$. By Lemma 4.3, $C^{\phi}_{X=x}$ is the set of all $u \in C$ such that either (i) $X(u) \neq x$, or (ii) $X(u)=x$ and $X=x$ is a weak cause of ϕ under u. As deciding (i) is polynomial, and deciding (ii) is in Σ_2^P, by Theorem 2.4, computing $C^{\phi}_{X=x}$ is in $\mathrm{FP}_{\parallel}^{\Sigma_2^P}$. Once $C^{\phi}_{X=x}$ is given, deciding (a) is possible with two Σ_2^P-oracle calls, by Theorem 4.1, and deciding (b) is polynomial. It is now well-known that two rounds of parallel Σ_2^P-oracle queries in a polynomial-time computation can be replaced by a single one [2]. Hence, the problem is in $\mathrm{P}_{\parallel}^{\Sigma_2^P}$.

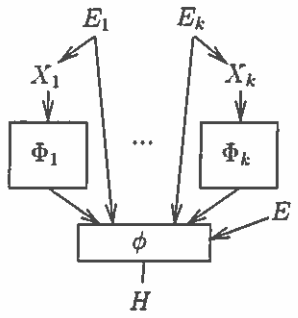


Figure 4: Schematic Construction for Evaluating k QBFs $\Phi_1,\ldots,\Phi_k$

Hardness for $\mathrm{P}_{\parallel}^{\Sigma_2^P}$ is shown by a reduction from deciding, given k QBFs $\Phi_i=\exists A_i \forall B_i \gamma_i$ with $i \in \{1,\ldots,k\}$, where each γ_i is a propositional formula on the variables $A_i=\{A_{i,1},\ldots,A_{i,m_i}\}$ and $B_i=\{B_{i,1},\ldots,B_{i,n_i}\}$, whether the number of valid formulas among $\Phi_1,\ldots,\Phi_k$ is even. W.l.o.g., $A_1 \cup B_1,\ldots,A_k \cup B_k$ are pairwise disjoint, Φ_1 is valid, and for each $j \in \{2,\ldots,k\}$, the validity of Φ_j implies the validity of Φ_{j-1} [41]. We create $M=(U,V,F)$, $X \subseteq V$, $x \in D(X)$, $\phi, C \subseteq D(U)$, P, and α such that $X=x$ is an α-partial explanation of ϕ relative to (C,P) iff the number of valid formulas among $\Phi_1,\ldots,\Phi_k$ is even. Roughly, the main idea behind this construction is as follows. For each Φ_i, we construct an instance of weak cause, that is, $M_i=(U_i,V_i,F_i)$, $X_i \subseteq V_i$, $x_i \in D(X_i)$, $u_i \in D(U_i)$ and an event ϕ_i, such that $X_i=x_i$ is a weak cause of ϕ_i under u_i in M_i iff Φ_i is valid. Then, M is the union of all M_i, enlarged by additional variables (see Fig. 4), and we define $X=X_1 \cup \cdots \cup X_k$ and $x=x_1 \ldots x_k$. By setting P to the uniform distribution over C and $\alpha=1/|C|$, we obtain that $X=x$ is an α-partial explanation of ϕ relative to (C,P), iff $X=x$ is an explanation of ϕ relative to $C^{\phi}_{X=x}$. The latter is made to hold iff the number of valid formulas among the Φ_i's is even. In detail, EX3 is violated, iff i is even, Φ_i is not valid, and Φ_{i-1} is valid. □

Theorem 4.5 *α-Partial Explanation Existence is Σ_3^P-complete.*

Proof. As for membership in Σ_3^P, by Theorem 4.4, deciding whether $X'=x'$ is an α-partial explanation of ϕ relative to (C,P) is in $\mathrm{P}_{\parallel}^{\Sigma_2^P}$. Thus, guessing some $X' \subseteq X$ and $x' \in D(X')$, and deciding whether $X'=x'$ is an α-partial explanation of ϕ relative to (C,P) is in Σ_3^P.

Σ_3^P-hardness is shown by a reduction from Explanation Existence. Given an instance of it, let P be the uniform distribution on C, and let $\alpha=1$. Then, $X'=x'$ is an α-partial explanation of ϕ relative to (C,P) iff $X'=x'$ is an explanation of ϕ relative to C. □

Theorem 4.6 *Partial Explanation is* $\mathrm{P}_{\|}^{\Sigma_2^{\mathrm{P}}}$*-complete.*

Proof (sketch). The membership part can be proved similarly as in the proof of Theorem 4.4. The hardness part follows easily from the hardness result in Theorem 4.4. □

Theorem 4.7 *Explanatory Power is* $\mathrm{FP}_{\|}^{\Sigma_2^{\mathrm{P}}}$*-complete.*

Proof (sketch). We compute first $\mathcal{C}_{X=x}^{\phi}$ and then $P(\mathcal{C}_{X=x}^{\phi} \mid X=x)$. By the proof of Theorem 4.4, the former is in $\mathrm{FP}_{\|}^{\Sigma_2^{\mathrm{P}}}$, while the latter can clearly be done in polynomial time. In summary, computing the explanatory power is in $\mathrm{FP}_{\|}^{\Sigma_2^{\mathrm{P}}}$.

$\mathrm{FP}_{\|}^{\Sigma_2^{\mathrm{P}}}$-hardness is shown by a reduction from computing, given k QBFs $\Phi_i = \exists A_i \forall B_i \gamma_i$ with $i \in \{1, \ldots, k\}$, where each γ_i is a propositional formula on the variables $A_i = \{A_{i,1}, \ldots, A_{i,m_i}\}$ and $B_i = \{B_{i,1}, \ldots, B_{i,n_i}\}$, the vector $(v_1, \ldots, v_k) \in \{0,1\}^k$ such that $v_i = 1$ iff Φ_i is valid, for all $i \in \{1, \ldots, k\}$. W.l.o.g., $A_1 \cup B_1, \ldots, A_k \cup B_k$ are pairwise disjoint, and Φ_1 is valid. Roughly, the main idea is to construct a problem instance such that $(v_1, \ldots, v_k)$ is the bitvector representation of the explanatory power of $X = x$. For each Φ_i, we construct $M_i = (U_i, V_i, F_i)$, $X_i \subseteq V_i$, $x_i \in D(X_i)$, $u_i \in D(U_i)$, and an event ϕ_i such that $X_i = x_i$ is a weak cause of ϕ_i under u_i in M_i iff Φ_i is valid. These models are then combined in M such that $u_i \in \mathcal{C}_{X=x}^{\phi}$ iff Φ_i is valid. Defining $P(u_i) = 2^{i-1}$ for all $i \in \{1, \ldots, k\}$ completes the reduction. □

4.3 SUCCINCT REPRESENTATION

Theorem 4.8 *Explanation is* Π_4^{P}*-complete in the case of succinct context sets.*

Proof (sketch). Recall that $X = x$ is an explanation of ϕ relative to $\mathcal{C}$ iff EX1–EX4 hold. Under succinct context sets, in EX1, deciding $\phi(u)$ for all $u \in \mathcal{C}$ is in co-NP. In EX4, deciding whether $X(u) = x$ and $X(u') \neq x$ hold for some $u, u' \in \mathcal{C}$ is in NP. By Theorem 2.4, deciding whether $X = x$ is a weak cause of ϕ under every $u \in \mathcal{C}$ with $X(u) = x$ in EX2 is in Π_3^{P}. Thus, deciding whether some $X' \subset X$ exists such that $X' = x | X'$ is a weak cause of ϕ under every $u \in \mathcal{C}$ with $X'(u) = x | X'$ is in Σ_4^{P}. That is, deciding EX3 is in Π_4^{P}. In summary, deciding whether EX1–EX4 hold is in Π_4^{P} under succinct context sets.

Hardness for Π_4^{P} is shown by a reduction from deciding whether a given QBF $\Phi = \forall A \exists B \forall C \exists D \gamma$ is valid, where γ is a propositional formula on the variables $A \cup B \cup C \cup D$. Roughly speaking, the main idea is to encode Φ in EX3, where the quantor "$\forall A$" is represented by considering all $X' \subset X$, the quantor "$\exists B$" is expressed by finding some $u \in D(U)$, and $\forall C \exists D \gamma$ is expressed by checking the complement of a weak cause. □

Theorem 4.9 *Partial Explanation is* Π_4^{P}*-complete in the case of succinct context sets.*

Proof (sketch). Membership in Π_4^{P} follows from Lemma 4.3 and Theorem 2.4. Hardness for Π_4^{P} can be proved similarly as in the proof of Theorem 4.8. □

5 RELATED WORK AND CONCLUSION

There is quite some work on algorithms and complexity of finding abductive explanations (e.g. [3, 6, 7, 8, 35, 37]) which play an important role in many AI problems including diagnosis, planning, or natural language processing. Roughly, a set of facts E is an abductive explanation of an observation O on some background theory T, if E is compatible with T and entails O; further minimality conditions are usually imposed on E. While causal and abductive explanations (in a standard logical setting [27, 37]) are apparently different concepts, they have similar complexity. In particular, deciding the existence of an abductive explanation is Σ_2^{P}-complete in the propositional context [8]; this matches our respective result on causal explanations for binary causal models. Computing causal and abductive explanations is polynomially intertranslatable in this case, while causal explanations from general causal models are harder to compute. Efficient transformations of causal into abductive explanations, and vice versa, is an interesting subject for further work.

Rather weakly related to ours are complexity results on maximum a posteriori explanations (MAPs, alias most probable explanations [25, 26]), which are a dominating notion of explanation in the probabilistic AI literature. Computing a MAP in a Bayesian belief network, i.e., an assignment to all variables given a partial assignment such that its probability is maximum, is NP-hard [39] but is feasible in polynomial time with an NP oracle. This result is quite different from our results on α-partial explanations, for two reasons: firstly, MAPs are computed from the set of all contexts, which is not part of the input. In this setting, α-partial explanations have higher complexity. Secondly, MAPs are *single contexts* which maximize probability for a given evidence, while α-partial explanations single out *subsets of contexts* which sensibly respect relevant information [20]. Computationally, it is more suitable to compare deciding $P(X = x) > 0$ in a belief network with our problem Partial Explanation under succinct context sets, where $\mathcal{C}$ contains all possible contexts and P emerges from independent exogenous variables. However, the former problem is NP-complete [5], while the latter is, by our results, Π_4^{P}-complete and thus much harder. We may expect a similar relationship for computing the explanatory power vs

the probability $P(X=x)$ in a belief network, which can be done in polynomial time with a #P oracle [34].

Our work on causal explanations continues and extends [11], and contributes in paving the way for efficient algorithms and implementations of the structural-model approach by Halpern and Pearl. Our results give a picture of the complexity of explanations in the general and the binary case. However, it remains to identify cases of lower complexity, and in particular islands of tractability. Meaningful restrictions must be found that eliminate several sources of complexity, which is not straightforward. This and refining the complexity picture is part of our ongoing work.

A APPENDIX: SELECTED PROOFS AND PROOF SKETCHES

A.1 PROOFS FOR SECTION 2

Proof of Lemma 2.3. ($\Rightarrow$) Assume that $X=x$ is a weak cause of ϕ under u. That is, (AC1) $X(u)=x$ and $\phi(u)$, and (AC2) some $W \subseteq V-X$, $\overline{x} \in D(X)$, $w \in D(W)$ exist such that (a) $\neg\phi_{\overline{x}w}(u)$ and (b) $\phi_{xw\hat{z}}(u)$ for all $\hat{Z} \subseteq V-(X \cup W)$ and $\hat{z}=\hat{Z}(u)$. In particular, $X'(u)=x'$ and $\phi(u)$. Moreover, as X_0 is no predecessor of any variable in ϕ, it follows that (a) $\neg\phi_{\overline{x}'w'}(u)$ and (b) $\phi_{x'w'\hat{z}}(u)$ for all $\hat{Z} \subseteq V-(X \cup W)$ and $\hat{z}=\hat{Z}(u)$, where $\overline{x}'=\overline{x}|X'$, $w'=wx_0$, and $x_0=x(X_0)$. This shows that $X'=x'$ is a weak cause of ϕ under u.

($\Leftarrow$) Assume that $X'=x'$ is a weak cause of ϕ under u. That is, (AC1) $X'(u)=x'$ and $\phi(u)$, and (AC2) some $W \subseteq V-X'$, $\overline{x}' \in D(X')$, $w \in D(W)$ exist such that (a) $\neg\phi_{\overline{x}'w}(u)$, and (b) $\phi_{x'w\hat{z}}(u)$ for all $\hat{Z} \subseteq V-(X' \cup W)$ and $\hat{z}=\hat{Z}(u)$. As $X_0(u)=x(X_0)$, it holds $X(u)=x$ and $\phi(u)$. Moreover, as X_0 is no predecessor of any variable in ϕ, it follows that (a) $\neg\phi_{\overline{x}'x_0w'}(u)$ and (b) $\phi_{x'x_0w'\hat{z}}(u)$ for all $\hat{Z} \subseteq V-(X \cup W)$ and $\hat{z}=\hat{Z}(u)$, where $w'=w|(W-\{X_0\})$, and $x_0=x(X_0)$. Hence, $X=x$ is a weak cause of ϕ under u. □

A.2 SELECTED PROOFS FOR SECTION 4

Proof of Theorem 4.1 (continued). For every $i \in \{1,2\}$, the causal model $M_i=(U,V_i,F_i)$ is defined by $U=\{E\}$ and $V_i=A_i \cup B_i \cup \{G,C_i\}$, where $D(S)=\{0,1,2\}$ for all $S \in B_i$, and $D(S)=\{0,1\}$ for all $S \in U_i \cup V_i - B_i$. Moreover, we define

$$\phi_i = (\gamma_i' \wedge \bigwedge_{S \in B_i} S \neq 2) \vee (C_i=0) \vee (G=1 \wedge C_i=1 \wedge \bigvee_{S \in B_i} S \neq 2),$$

where γ_i' is obtained from γ_i by replacing each $S \in A_i \cup B_i$ by "$S=1$". The functions in $F_i = \{F_S^i \mid S \in V_i\}$ are defined as follows:

- $F_S^i=0$ for all $S \in A_i \cup \{G,C_i\}$,
- $F_S^i=G+C_i$ for all $S \in B_i$.

As shown in [11, 12], for every $i \in \{1,2\}$ and $u \in D(U)$, it holds that $G=0$ is a weak cause of ϕ_i under u in M_i iff Φ_i is valid.

The causal model $M=(U,V,F)$ is now defined by $V=V_1 \cup V_2 \cup \{G',H\}$ and $F=F_1 \cup F_2 \cup \{F_{G'}=E,\ F_H=1$ iff $(E=0 \wedge \phi_1) \vee (E=1 \wedge \phi_2)$ is true$\}$. Let ϕ be defined as $H=1$, and let $u_1,u_2 \in D(U)$ be defined by $u_1(E)=0$ and $u_2(E)=1$. Observe that ϕ is primitive.

For every $i \in \{1,2\}$ and $u \in D(U)$, it holds that $G=0$ is a weak cause of ϕ_i under u in M iff Φ_i is valid. Hence, for every $i \in \{1,2\}$,

(i) $G=0$ is a weak cause of ϕ under u_i in M iff Φ_i is valid.

By Lemma 2.3, the following statements hold:

(ii) $G=0$ is a weak cause of ϕ under u_1 in M iff $G=0 \wedge G'=0$ is a weak cause of ϕ under u_1 in M.

(iii) $G'=0$ is not a weak cause of ϕ under u_1 in M.

Using these results, we now show that $G=0 \wedge G'=0$ is an explanation of ϕ relative to $\mathcal{C}=\{u_1,u_2\}$ iff Φ_1 is valid and Φ_2 is not valid.

($\Rightarrow$) Assume that $G=0 \wedge G'=0$ is an explanation of ϕ relative to $\mathcal{C}$. In particular, by EX2, $G=0 \wedge G'=0$ is a weak cause of ϕ under u_1. Moreover, by EX3, $G=0$ is either not a weak cause of ϕ under u_1, or not a weak cause of ϕ under u_2. By (ii), $G=0$ is a weak cause of ϕ under u_1. Hence, $G=0$ is not a weak cause of ϕ under u_2. By (i), Φ_1 is valid, and Φ_2 is not valid.

($\Leftarrow$) Assume that Φ_1 is valid and Φ_2 is not valid. We first show that EX1 holds. As $C_i(u)=0$ for all $i \in \{1,2\}$ and $u \in \mathcal{C}$, we get $\phi_i(u)$ for all $i \in \{1,2\}$ and $u \in \mathcal{C}$. Thus, $\phi(u)$ for all $u \in \mathcal{C}$. To see that EX4 holds, observe that $G(u_1)=G'(u_1)=0$, while $G(u_2)=0$ and $G'(u_2)=1$. We next show that EX2 holds. By (i), $G=0$ is a weak cause of ϕ under u_1. By (ii), it follows that $G=0 \wedge G'=0$ is a weak cause of ϕ under u_1. We now show that EX3 holds. By (i), $G=0$ is not a weak cause of ϕ under u_2. By (iii), $G'=0$ is not a weak cause of ϕ under u_1. □

Proof of Theorem 4.2 (continued). Hardness for Σ_3^P is shown by a reduction from deciding whether a given QBF $\Phi=\exists B \forall C \exists D\, \gamma$ is valid, where γ is a propositional formula on the variables $B=\{B_1,\ldots,B_l\}$, $C=\{C_1,\ldots,$

$C_m\}$, and $D=\{D_1,\ldots,D_n\}$. We build $M=(U,V,F)$, $X\subseteq V$, $C\subseteq D(U)$, and ϕ as required such that Φ is valid iff some $X'\subseteq X$ and $x'\in D(X')$ exist such that $X'=x'$ is an explanation of ϕ relative to C.

We define $U=\{I,U_0,U_0',\ldots,U_k,U_k'\}$, where $D(I)=\{0,\ldots,l+1\}$ and $D(S)=\{0,1\}$ for all $S\in U-\{I\}$. Let $C=\{u_0,u_0',\ldots,u_l,u_l',u_{l+1}\}$, where u_i (resp., u_i') is the unique $u\in D(U)$ such that $\varepsilon_i(u)$ (resp., $\varepsilon_i'(u)$) holds, and ε_i (resp., ε_i') for every $i\in\{0,\ldots,l+1\}$ (resp., $i\in\{0,\ldots,l\}$) is defined by:

$$\begin{aligned}\varepsilon_i &= I=i\wedge U_0=0\wedge U_0'=1\\ &\quad\wedge\bigwedge_{i=1}^{l}(U_i=0\wedge U_i'=0),\\ \varepsilon_i' &= I=i\wedge U_0=0\wedge U_0'=0\\ &\quad\wedge\bigwedge_{i=1}^{l}(U_i=0\wedge U_i'=0).\end{aligned}$$

Define $M=(U,V,F)$ as follows. Let $V=B\cup B'\cup C\cup D\cup\{X_0,X_0',E,E',Y\}$, where $B'=\{B_1',\ldots,B_l'\}$, $D(S)=\{0,1,2\}$ for all $S\in D$, and $D(S)=\{0,1\}$ for all $S\in V-D$. Let

$$\begin{aligned}\alpha &= (\neg\gamma'\wedge\bigwedge_{S\in D}S\neq 2)\vee(E=0)\\ &\quad\vee(X_0=1\wedge E=1\wedge\bigvee_{S\in D}S\neq 2),\\ \phi_1' &= (\varepsilon_0\vee\varepsilon_0'\rightarrow(X_0=0\wedge\bigwedge_{i=1}^{l}B_i\neq B_i')\\ &\quad\vee(\bigvee_{i=1}^{l}(B_i=1\wedge B_i'=1))\vee E'=0),\\ \phi_2' &= \bigwedge_{i=1}^{l}(\varepsilon_i\vee\varepsilon_i'\rightarrow B_i=0\vee B_i'=0),\\ \phi_3' &= (\varepsilon_{l+1}\rightarrow(\alpha\wedge\bigwedge_{i=1}^{l}B_i\neq B_i')\\ &\quad\vee(\bigvee_{i=1}^{l}(B_i=1\wedge B_i'=1))\vee E'=0),\end{aligned}$$

where γ' is obtained from γ by replacing each $S\in B\cup C\cup D$ by "$S=1$". We are now ready to define the functions $F=\{F_S\mid S\in V\}$ as follows:

- $F_{B_i}=U_i$ and $F_{B_i'}=U_i'$ for all $i\in\{1,\ldots,l\}$,
- $F_{X_0}=U_0$ and $F_{X_0'}=U_0'$,
- $F_S=0$ for all $S\in C\cup\{E,E'\}$,
- $F_S=X_0+E$ for all $S\in D$,
- $F_Y=1$ iff $\phi_1'\vee\phi_2'\vee\phi_3'$ is true.

Define $X=B\cup B'\cup\{X_0,X_0'\}$, and let ϕ be $Y=1$. Notice that ϕ is primitive.

For every truth assignment τ to the variables in B, we use $[B/\tau(B)]$ to denote the substitution $[B_1/\tau(B_1),\ldots,B_l/\tau(B_l)]$, and we define $\alpha^\tau=\alpha[B/\tau(B)]$. Define $x_0=0$, and let $u\in D(U)$ with $X_0(u)=x_0$. Then, $X_0=x_0$ is a weak cause of α^τ under u iff $\exists C\,\forall D\,\neg\gamma[B/\tau(B)]$ is valid [11, 12]. That is, $X_0=x_0$ is not a weak cause of α^τ under u iff $\forall C\,\exists D\,\gamma[B/\tau(B)]$ is valid. Thus, Lemma 2.3 implies the following fact:

(*) For every $X'\subseteq B\cup B'\cup\{X_0,X_0'\}$ with $X_0\in X'$, it holds that $X'=X'(u)$ is not a weak cause of α^τ under u iff $\forall C\,\exists D\,\gamma[B/\tau(B)]$ is valid.

Using this result, it can be shown that Φ is valid iff some $X'\subseteq X$ and $x'\in D(X')$ exist such that $X'=x'$ is an explanation of ϕ relative to C (see [10] for details). □

Proof of Lemma 4.3. Clearly, $\mathcal{C}^{\phi}_{X=x}$ does not contain any $u\in\mathcal{C}$ such that $X(u)=x$ and that $X=x$ is not a weak cause of ϕ under u, as otherwise EX2 would be violated. Hence, $\mathcal{C}^{\phi}_{X=x}$ is a subset of the set of all $u\in\mathcal{C}$ such that either (i) or (ii). Assume now that some $u'\in\mathcal{C}$ with $X(u')\neq x$ does not belong to $\mathcal{C}^{\phi}_{X=x}$. Then, $X=x$ is an explanation of ϕ relative to $\mathcal{C}'=\mathcal{C}^{\phi}_{X=x}\cup\{u'\}$. But this contradicts $\mathcal{C}^{\phi}_{X=x}$ being the largest such $\mathcal{C}'$. Assume next that some $u'\in\mathcal{C}$ such that $X(u')=x$ and that $X=x$ is a weak cause of ϕ under u' does not belong to $\mathcal{C}^{\phi}_{X=x}$. Then, $X=x$ is an explanation of ϕ relative to $\mathcal{C}'=\mathcal{C}^{\phi}_{X=x}\cup\{u'\}$. But this contradicts again $\mathcal{C}^{\phi}_{X=x}$ being the largest such $\mathcal{C}'$. Hence, $\mathcal{C}^{\phi}_{X=x}$ is the set of all $u\in\mathcal{C}$ such that either (i) or (ii). □

Proof of Theorem 4.4 (continued). We construct $M=(U,V,F)$, $X\subseteq V$, $x\in D(X)$, ϕ, $\mathcal{C}\subseteq D(U)$, P, and α as required, such that $X=x$ is an α-partial explanation of ϕ relative to $(\mathcal{C},P)$ iff the number of valid formulas among $\Phi_1,\ldots,\Phi_k$ is even.

For $i\in\{1,\ldots,k\}$, define the causal models $M_i=(U_i,V_i,F_i)$ as follows. The exogenous and endogenous variables are defined by $U_i=\{E_i\}$ and $V_i=A_i\cup B_i\cup\{C_i,G_i\}$, respectively. Define $D(S)=\{0,1,2\}$ for all $S\in B_i$, and $D(S)=\{0,1\}$ for all $S\in U_i\cup V_i-B_i$. We define

$$\begin{aligned}\phi_i &= (\gamma_i'\wedge\bigwedge_{S\in B_i}S\neq 2)\vee(C_i=0)\\ &\quad\vee(G_i=1\wedge C_i=1\wedge\bigvee_{S\in B_i}S\neq 2),\end{aligned}$$

where γ_i' is obtained from γ_i by replacing each $S\in A_i\cup B_i$ by "$S=1$". The functions in $F_i=\{F_S^i\mid S\in V_i\}$ are defined as follows:

- $F^i_{G_i}=E_i$,

- $F^i_S = 0$ for all $S \in \{C_i\} \cup A_i$,
- $F^i_S = G_i + C_i$ for all $S \in B_i$,

For each $i \in \{1,\ldots,k\}$, let $X_i = \{G_i\}$, and define $x_i \in D(X_i)$ and $u_i \in D(U_i)$ by $x_i(G_i) = 0$ and $u_i(E_i) = 0$. Then, for every $i \in \{1,\ldots,k\}$, $X_i = x_i$ is a weak cause of ϕ_i under u_i in M_i iff Φ_i is valid (the construction is similar as in the proof of Theorem 4.1, the only difference is that we have $F^i_{G_i} = E_i$ here, instead of $F^i_{G_i} = 0$). Observe also that $\phi_i(u)$ holds for all $u \in D(U_i)$.

Define the causal model $M = (U, V, F)$ by $U = U_1 \cup \cdots \cup U_k \cup \{E\}$, where $D(E) = \{0,\ldots,k\}$, $V = V_1 \cup \cdots \cup V_k \cup \{H\}$, and $F = F_1 \cup \cdots \cup F_k \cup \{F_H\}$, where $F_H = 1$ iff

$$\Big(\bigwedge_{i\in\{1,\ldots,k\}} \varepsilon_i \rightarrow \phi_i\Big) \wedge \Big(\bigwedge_{i\in\{1,\ldots,k\},\, i \text{ even}} \varepsilon'_i \rightarrow \phi_{i-1}\Big) \wedge \Big(\bigwedge_{i\in\{1,\ldots,k\},\, i \text{ odd}} \varepsilon'_i \rightarrow \top\Big)$$

is true, and ε_i and ε'_i are defined as follows for every $i \in \{1,\ldots,k\}$:

$$\varepsilon_i = (E = i) \wedge \Big(\bigwedge_{j\in\{1,\ldots,k\}} (E_j = 0)\Big),$$

$$\varepsilon'_i = (E = 0) \wedge (E_i = 1) \wedge \Big(\bigwedge_{j\in\{1,\ldots,k\}-\{i\}} (E_j = 0)\Big).$$

For every $i \in \{1,\ldots,k\}$, let u_i (resp., u'_i) be the unique $u \in D(U)$ such that $\varepsilon_i(u)$ (resp., $\varepsilon'_i(u)$). Let $Y = \{H\}$, and let ϕ be $Y = 1$. We define $\mathcal{C} = \{u_1,\ldots,u_k,u'_1,\ldots,u'_k\}$, $P(u) = 1/2k$ for all $u \in \mathcal{C}$, and $\alpha = 1/2k$. Define $X = \{G_1,\ldots,G_k\}$ and $x = x_1 \cdots x_k$ $(= 0 \cdots 0)$.

Observe that ϕ is primitive, P is the uniform distribution over $\mathcal{C}$, and $\phi(u)$ for all $u \in \mathcal{C}$. By Lemma 2.3, the following holds for all $i \in \{1,\ldots,k\}$, all $X' \subseteq X$, and $x' = x|X'$:

(i) If $X_i \subseteq X'$, then $X' = x'$ is a weak cause of ϕ under u_i iff Φ_i is valid.

(ii) If i is even and $X_{i-1} \subseteq X'$, then $X' = x'$ is a weak cause of ϕ under u'_i iff Φ_{i-1} is valid.

(iii) If i is odd, then $X' = x'$ is not a weak cause of ϕ under u'_i.

(iv) If $X_i \not\subseteq X'$, then $X' = x'$ is not a weak cause of ϕ under u_i.

By Lemma 4.3, $\mathcal{C}^{\phi}_{X=x}$ is the set of all $u \in \mathcal{C}$ such that either (a) $X(u) \neq x$, or (b) $X(u) = x$ and $X = x$ is a weak cause of ϕ under u. By (i), $\mathcal{C}^{\phi}_{X=x} = \{u'_1,\ldots,u'_k\} \cup \{u_i \mid i \in \{1,\ldots,k\},\ \Phi_i \text{ is valid}\}$. It can now be shown that $X = x$ is an α-partial explanation of ϕ relative to $(\mathcal{C}, P)$ iff the number of valid formulas among $\Phi_1,\ldots,\Phi_k$ is even (see [10] for details). □

Proof of Theorem 4.8 (continued). Hardness for Π^{P}_4 is shown by a reduction from the Π^{P}_4-complete problem of deciding whether a given QBF $\Phi = \forall A\, \exists B\, \forall C\, \exists D\, \gamma$ is valid, where γ is a propositional formula on the variables $A = \{A_1,\ldots,A_k\}$, $B = \{B_1,\ldots,B_l\}$, $C = \{C_1,\ldots,C_m\}$, and $D = \{D_1,\ldots,D_n\}$. We build $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, $\mathcal{C} \subseteq D(U)$, and ϕ as in the statement of the theorem such that $X = x$ is an explanation of ϕ relative to $\mathcal{C}$ iff Φ is valid.

We define the exogenous variables by $U = B \cup \{U_0, U_1, U_1', \ldots, U_k, U_k'\}$, where $D(S) = \{0,1\}$ for all $S \in U$. Define the set of contexts by $\mathcal{C} = \{u \in D(U) \mid (\varepsilon_0 \vee \varepsilon_1 \vee \varepsilon_2)(u)\}$, where:

$$\varepsilon_0 = U_0{=}0 \wedge \bigwedge_{i=1}^{k} (U_i{=}0 \wedge U_i'{=}0),$$

$$\varepsilon_1 = U_0{=}0 \wedge \bigvee_{i=1}^{k} \big(((U_i{=}1 \wedge U_i'{=}0) \vee (U_i{=}0 \wedge U_i'{=}1)) \wedge \bigwedge_{j\in\{1,\ldots,k\}-\{i\}} (U_j{=}0 \wedge U_j'{=}0)\big),$$

$$\varepsilon_2 = U_0{=}1 \vee \bigvee_{i=1}^{k} (U_i{=}1 \wedge U_i'{=}1).$$

We define $M = (U, V, F)$ as follows. Define $V = A \cup A' \cup C \cup D \cup \{X_0, E, F, Y\}$, where $A' = \{A_1', \ldots, A_k'\}$, $D(S) = \{0,1,2\}$ for all $S \in D$, and $D(S) = \{0,1\}$ for all $S \in V - D$. Let

$$\alpha = (\neg\gamma' \wedge \bigwedge_{S\in D} S \neq 2) \vee (E = 0) \vee (X_0 = 1 \wedge E = 1 \wedge \bigvee_{S\in D} S \neq 2),$$

$$\phi' = (\varepsilon_0 \rightarrow X_0 = 0) \wedge (\varepsilon_2 \rightarrow \top) \wedge (\varepsilon_1 \rightarrow (\alpha \wedge \bigwedge_{i=1}^{k} A_i \neq A_i') \vee (\bigvee_{i=1}^{k} (A_i = 1 \wedge A_i' = 1)) \vee F = 0),$$

where γ' is obtained from γ by replacing each $S \in A \cup B \cup C \cup D$ by "$S = 1$". We are now ready to define the functions $F = \{F_S \mid S \in V\}$ as follows:

- $F_{A_i} = U_i$ and $F_{A_i'} = U_i'$ for all $i \in \{1,\ldots,k\}$,
- $F_{X_0} = U_0$, and $F_S = 0$ for all $S \in C \cup \{E, F\}$,
- $F_S = X_0 + E$ for all $S \in D$,
- $F_Y = 1$ iff ϕ' is true.

Let $X = A \cup A' \cup \{X_0\}$, and let $x \in D(X)$ be given by $x(S) = 0$ for all $S \in X$. Let ϕ be $Y = 1$. Notice that ϕ is primitive. It can now be shown that Φ is valid iff $X = x$ is an explanation of ϕ relative to C (see [10] for details). □

Acknowledgments

This work has been partially supported by the Austrian Science Fund Project Z29-INF, a DFG grant, and a Marie Curie Individual Fellowship of the European Community. We are grateful to the referees for their useful comments.

References

[1] A. Balke and J. Pearl. Probabilistic evaluation of counterfactual queries. In *Proc. AAAI-94*, pp. 230–237, 1994.

[2] S. Buss and L. Hay. On truth-table reducibility to SAT. *Inf. Comput.*, 91:86–102, 1991.

[3] T. Bylander, D. Allemang, M. C. Tanner, and J. R. Josephson. The computational complexity of abduction. *Artif. Intell.*, 49:25–60, 1991.

[4] U. Chajewska and J. Y. Halpern. Defining explanation in probabilistic systems. In *Proc. UAI-97*, pp. 62–71, 1997.

[5] G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artif. Intell.*, 42:393–405, 1990.

[6] A. del Val. On some tractable classes in deduction and abduction. *Artif. Intell.*, 116(1–2):297–313, 2000.

[7] A. del Val. The complexity of restricted consequence finding and abduction. In *Proc. AAAI/IAAI-00*, pp. 337–342, 2001.

[8] T. Eiter and G. Gottlob. The complexity of logic-based abduction. *J. ACM*, 42(1):3–42, January 1995.

[9] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. The KR System dlv: Progress report, comparisons, and benchmarks. In *Proc. KR-98*, pp. 406–417, 1998.

[10] T. Eiter and T. Lukasiewicz. Complexity results for explanations in the structural-model approach. Technical Report INFSYS RR-1843-01-08, Institut für Informationssysteme, TU Wien, 2001.

[11] T. Eiter and T. Lukasiewicz. Complexity results for structure-based causality. In *Proc. IJCAI-01*, pp. 35–40, 2001.

[12] T. Eiter and T. Lukasiewicz. Complexity results for structure-based causality. Technical Report INFSYS RR-1843-01-01, Institut für Informationssysteme, TU Wien, 2001.

[13] D. Galles and J. Pearl. Axioms of causal relevance. *Artif. Intell.*, 97:9–43, 1997.

[14] P. Gärdenfors. *Knowledge in Flux*. MIT Press, 1988.

[15] H. Geffner. Causal theories for nonmonotonic reasoning. In *Proc. AAAI-90*, pp. 524–530, 1990.

[16] H. Geffner. *Default Reasoning: Causal and Conditional Theories*. MIT Press, 1992.

[17] J. Y. Halpern. Axiomatizing causal reasoning. *J. Artif. Intell. Res.*, 12:317–337, 2000.

[18] J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach. Technical Report R-266, UCLA Cognitive Systems Lab, 2000.

[19] J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach – Part I: Causes. In *Proc. UAI-01*, pp. 194–202, 2001.

[20] J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach – Part II: Explanations. In *Proc. IJCAI-01*, pp. 27–34, 2001.

[21] C. G. Hempel. *Aspects of Scientific Explanation*. Free Press, 1965.

[22] M. Henrion and M. J. Druzdzel. Qualitative propagation and scenario-based approaches to explanation of probabilistic reasoning. In *Uncertainty in Artificial Intelligence 6*, pp. 17–32. Elsevier Science, 1990.

[23] B. Jenner and J. Toran. Computing functions with parallel queries to NP. *Theor. Comput. Sci.*, 141:175–193, 1995.

[24] D. S. Johnson. A catalog of complexity classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 2. Elsevier Science, 1990.

[25] J.Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[26] K. Kask and R. Dechter. A general scheme for automatic generation of search heuristics from specification dependencies. *Artif. Intell.*, 129:91–131, 2001.

[27] K. Konolige. Abduction versus closure in causal theories. *Artif. Intell.*, 53:255–272, 1992.

[28] V. Lifschitz. On the logic of causal explanation. *Artif. Intell.*, 96:451–465, 1997.

[29] N. McCain and H. Turner. Causal theories of action and change. In *Proc. AAAI-97*, pp. 460–465, 1997.

[30] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[31] J. Pearl. Reasoning with cause and effect. In *Proc. IJCAI-99*, pp. 1437–1449, 1999.

[32] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[33] J. Rintanen. Constructing conditional plans by a theorem-prover. *J. Artif. Intell. Res.*, 10:323–352, 1999.

[34] D. Roth. On the hardness of approximate reasoning. *Artif. Intell.*, 82(1–2):273–302, 1996.

[35] V. Rutenburg. Propositional truth maintenance systems: Classification and complexity analysis. *Ann. Math. Artif. Intell.*, 10:207–231, 1994.

[36] W. C. Salmon. *Four Decades of Scientific Explanation*. University of Minnesota Press, 1989.

[37] B. Selman and H. J. Levesque. Support set selection for abductive and default reasoning. *Artif. Intell.*, 82:259–272, 1996.

[38] S. E. Shimony. Explanation, irrelevance, and statistical independence. In *Proc. AAAI-91*, pp. 482–487, 1991.

[39] S. E. Shimony. Finding MAPs for belief networks is NP-hard. *Artif. Intell.*, 68:399–410, 1994.

[40] H. Turner. A logic of universal causation. *Artif. Intell.*, 113:87–123, 1999.

[41] K. Wagner. Bounded query classes. *SIAM Journal of Computing*, 19(5):833–846, 1990.

Complexity Results for Paraconsistent Inference Relations

Sylvie Coste-Marquis
CRIL/Université d'Artois
rue de l'Université - S.P. 16
62307 Lens Cedex - France
coste@cril.univ-artois.fr

Pierre Marquis
CRIL/Université d'Artois
rue de l'Université - S.P. 16
62307 Lens Cedex - France
marquis@cril.univ-artois.fr

Abstract

In this paper, the complexity of several paraconsistent inference relations, based on multivalued logics, is investigated. Many inference relations pointed out so far by Arieli and Avron, Besnard and Schaub, D'Ottaviano and da Costa, Frisch, Levesque, Priest are considered from the computational side. All these relations can be gathered into two categories: the basic ones stem directly from the notions of models within 3 or 4-valued logics, while the refined ones are based on notions of preferred models for such logics. Completing complexity results by Cadoli and Schaerf (centered on the basic relations), we show that the refined paraconsistent inference relations that have been defined in the framework of multivalued logics are highly intractable. Especially, we prove that the inference problems corresponding to these relations are Π_2^p-complete, even in the simple case where the database is a CNF formula and the query is a propositional symbol.

1 INTRODUCTION

While classical logic is considered as a base line in many AI studies, it is acknowledged that classical entailment cannot be used as such to model commonsense inference. Among its major drawbacks is its inability to handle inconsistent information; indeed, in presence of a contradiction, every formula can be derived (this is the well-known *ex falso quodlibet sequitur* of classical logic).

Such a trivialization problem is very significant both from the theoretical side and from the practical side. Specifically, it is strongly connected to the possibility to handle exceptions (another salient aspect of common-sense reasoning). From the practical side, its importance comes from the fact that many actual, large-sized databases are inconsistent (specifically when they integrate information stemming from multiple sources).

Many logics have been developed so far in order to avoid trivialization in presence of inconsistency. They are called *paraconsistent logics*. Both the complexity of the problem and its significance are reflected by the number of approaches to paraconsistent reasoning that can be found in the literature (see [Besnard and Hunter, 1998; Hunter, 1998] for a survey). Indeed, paraconsistency can be achieved in various ways, mainly:

- by *restricting the proof theory* of classical logic so as to retain only a subset of the classical proofs as admissible (e.g., this is what is done in C_ω logic [da Costa, 1974] and in quasi-classical logic [Besnard and Hunter, 1995; Hunter, 2000]).
- by *focusing on the consistent subsets* of the inconsistent database, eventually restricted to the most preferred ones, when some preferential information can be exploited [Rescher and Manor, 1970; Fagin *et al.*, 1983; Ginsberg, 1986; Baral *et al.*, 1991; Pinkas and Loui, 1992; Benferhat *et al.*, 1993]. Such techniques are closely related to so-called syntax-based approaches to belief revision [Nebel, 1992; Nebel, 1994] and to the framework for supernormal default reasoning with priorities from [Brewka, 1989].
- by associating to each infered formula a justification under the form of a subset of the information used to derive it, and by reasoning on such arguments whenever some mutually inconsistent formulas can be derived. This is the basic idea of *argumentative logics*, see e.g., [Dung, 1995;

Elvang-Goransson and Hunter, 1995; Bondarenko *et al.*, 1997].

- by *merging* the various pieces of belief from the inconsistent database, see e.g., [Lin, 1996; Revesz, 1997; Konieczny and Pino Pérez, 1998; Konieczny, 2000].
- by preventing inconsistent databases from having no model, through the consideration of more general notions of interpretations (or worlds). Several *multivalued logics* are related to this line of research [D'Ottaviano and da Costa, 1970; Belnap, 1977; Frisch, 1987; Levesque, 1989; Priest, 1989; Priest, 1991; Besnard and Schaub, 1997; Besnard and Schaub, 1998; Arieli and Avron, 1998].

Of course, these approaches are not mutually exclusive. For instance, some merging operators and argument selection policies are based on consistent subsets, the set of inference rules used in the proof theory of some multivalued logics is a subset of those used in classical logic, and conversely some logics based on a subclassical proof theory can be given some multivalued semantics.

In this paper, we mainly focus on the *multivalued logic* approach to inconsistency handling. Compared with other approaches to inconsistent-tolerant reasoning, like forms of belief merging and the maxcons approach (i.e., the technique based on the selection of *max*imally *cons*istent subsets of the database) multivalued paraconsistent logics prevent inference from trivializing, even in the restricted case where the database consists of a single inconsistent formula.

In the following, we specifically consider some four-valued logics and three-valued logics proposed so far, including Arieli and Avron's *FOUR* logic [Arieli and Avron, 1998] (based on Belnap's work [Belnap, 1977]), Besnard and Schaub's one [Besnard and Schaub, 1997], D'Ottaviano and da Costa's J_3 logic [D'Ottaviano and da Costa, 1970], and several restrictions of them, especially Priest's LP and LP_m logics [Priest, 1989; Priest, 1991], Frisch's RP logic [Frisch, 1987], Levesque's logic of 3-inference [Levesque, 1989]. We also consider some paraconsistent relations given by Besnard and Schaub [Besnard and Schaub, 1998]. All these relations can be gathered into two categories: the basic ones stem directly from the notions of models within 3 or 4-valued logics, while the refined ones are based on notions of preferred models for such logics.

Many of these inference relations have been investigated in depth from a logical point of view (see e.g., [Arieli and Avron, 1998]), but the study of their computational complexity aspects is far less complete. The notable exception concerns some of the basic relations, the complexity of which has been analyzed by Levesque [Levesque, 1989] and by Cadoli and Schaerf [Cadoli and Schaerf, 1996]. This paper contributes to fill the gap. For each inference relation under consideration, the complexity of the corresponding decision problem is identified, in the general case (i.e., without any restriction over the database Σ or the query γ), and in some restricted cases (specifically, when Σ is a CNF formula and γ is a propositional symbol).

From our complexity analysis, the following conclusions can be drawn. Firstly, the inference problem for the basic relations considered in this paper (i.e., Arieli and Avron's $\models^4$, D'Ottaviano and da Costa's $\models^3$ and its restrictions: Priest's $\models_{LP}$, Frisch's $\models_{RP}$ and Levesque's $\models_L$) is coNP-complete in the general case (and even in P in some restricted situations). This is not so high, especially when compared with the complexity of the inference problems associated to other approaches to inconsistency-tolerant inference, like the maxcons ones. Unfortunately, such basic inference relations are typically too cautious. For instance, the disjunctive syllogism is not satisfied by any of them. As a consequence, none of them coincides with classical entailment in the restricted situation where Σ is a (classically) consistent CNF formula. In order to design less cautious inference relations, it is sufficient to focus on some preferred models, e.g., those which are as close as possible to the models of classical logic. This is the key idea underlying the inference relations from the second family considered in this paper, i.e., the refined ones. Secondly, we have established that the inference problem for the refined relations under consideration (i.e., Arieli and Avron's $\models^4_{I_1}$ and $\models^4_{I_2}$ [Arieli and Avron, 1998], Priest's $\models_{LP_m}$ [Priest, 1989; Priest, 1991], Besnard and Schaub $\models_{BS}$ [Besnard and Schaub, 1997]) as well as Besnard and Schaub's $\vdash_s$ and $\vdash^{\pm}_s$ [Besnard and Schaub, 1998] is Π^p_2-complete, even in the restricted case where the database Σ is a CNF formula and the query γ is a propositional symbol. Such an increase in computational complexity seems to be the price to be paid for designing paraconsistent relations that are not too cautious in the framework of multivalued logics.

The rest of this paper is organized as follows. In Section 2, the multivalued logics considered in the paper are described, both from the syntactical side and from the semantical one. On this ground, several paraconsistent inference relations are given in Section 3. The complexity results corresponding to their decision problems are reported in Section 4. Some concluding remarks are provided in Section 5. Proof sketches are reported in an appendix.

2 MULTIVALUED LOGICS

2.1 SYNTACTICAL ASPECTS

In the following, the propositional language $PROP^4_{PS}$, and its restrictions $PROP^3_{PS}$ and $PROP^2_{PS}$ are considered. They are based on a finite set PS of propositional symbols.

Definition 1 ($PROP^4_{PS}$, $PROP^3_{PS}$ and $PROP^2_{PS}$)

- *$PROP^4_{PS}$ is the propositional language over PS generated from the constant symbols true, false, both, unknown, and the connectives $\neg$, $\vee$, $\wedge$, $\supset$, $\oplus$, and $\otimes$.*
- *$PROP^3_{PS}$ is the propositional language over PS generated from the constant symbols true, false, both, and the connectives $\neg$, $\vee$, $\wedge$, $\supset$, and $\oplus$.*
- *$PROP^2_{PS}$ is the propositional language over PS generated from the constant symbols true and false, and the connectives $\neg$, $\vee$, $\wedge$, and $\supset$.*

Clearly enough, the fragment $PROP^2_{PS}$ of $PROP^4_{PS}$ coincides with a standard language for classical propositional logic. A restricted fragment of it gathers the formula generated over PS using the connectives $\neg$, $\wedge$, $\vee$, only, which is referred to as $\{\neg, \wedge, \vee\}$ *fragment*. A proper subset of this fragment is composed by the *CNF formulas*, i.e., the (finite) conjunctions of clauses, where a clause is a (finite) disjunction of literals (the symbols from PS, eventually negated).

2.2 SEMANTICAL ASPECTS

Let us now explain how the formulas of $PROP^4_{PS}$ and its subsets can be interpreted. Obviously enough, in multivalued logics, there are more truth values than just 0 (false) and 1 (true):

Definition 2 (Interpretations)
A 4-interpretation (resp. a 3-interpretation, a 2-interpretation) over PS is a total function I from PS to $FOUR = \{0, 1, \top, \bot\}$ (resp. $THREE = \{0, 1, \top\}$, $TWO = \{0, 1\}$).

Here, $\bot$ intuitively denotes lack of information while $\top$ indicates inconsistency. All the connectives under consideration in this paper are *truth functional* ones. For every 4-interpretation I over PS, we define $I(true) = 1$, $I(false) = 0$, $I(both) = \top$, $I(unknown) = \bot$. The semantics $I(\phi)$ of a formula ϕ from $PROP^4_{PS}$ in I is defined compositionally in the obvious way, given the truth tables reported in Table 1.

The semantics of a formula from $PROP^3_{PS}$ (resp. $PROP^2_{PS}$) in a 3-interpretation (resp. a 2-interpretation) is defined in the obvious compositional way, by considering the reductions of the previous truth tables to $THREE$ (resp. TWO). This explains why *unknown* and the consensus operator $\otimes$ are not considered as connectives in $PROP^3_{PS}$ (just like *unknown*, *both*, $\otimes$, and the gullability operator $\oplus$ are not considered as connectives in $PROP^2_{PS}$), since the set of truth values must be closed under the connectives.

Unlike other multivalued logics, the set of *designated values* considered in $FOUR$ and its restriction $THREE$ is $\{1, \top\}$. This leads to the following notions of *models*:

Definition 3 (Models)
Let I be a 4-interpretation (resp. a 3-interpretation, a 2-interpretation) I over PS and ϕ be a formula from $PROP^4_{PS}$ (resp. $PROP^3_{PS}$, $PROP^2_{PS}$). I is a 4-model *(resp. a 3-model, a 2-model) of ϕ iff $I(\phi) \in \{1, \top\}$.*

Example 1 *The 4-interpretation I given by $I(a) = \top$ and $I(x) = 1$ for all $x \in PS \setminus \{a\}$ is a 4-model of $\Sigma = a \wedge \neg a \wedge b$ since $I(\Sigma) = \top$. I can also be viewed as a 3-model of Σ. Note that the fact that Σ has no 2-models does not prevent it from having some 3-models or 4-models.*

On this ground, two formulas ϕ and ψ from $PROP^4_{PS}$ (resp. $PROP^3_{PS}$, $PROP^2_{PS}$) are said to be *equivalent*, noted $\phi \equiv^4 \psi$ (resp. $\phi \equiv^3 \psi$, $\phi \equiv^2 \psi$) iff they have the same set of 4-models (resp. 3-models, 2-models).

Once this is stated, two remarks can be done. On the one hand, some of the connectives could easily be defined from others while preserving equivalence; for instance, this is the case for $\vee$ since $\phi \vee \psi \equiv^4 \neg(\neg\phi \wedge \neg\psi)$ (i.e., De Morgan's laws are satisfied in $FOUR$, hence in $THREE$); another example is $\oplus$ which can be defined from *both*, $\vee$ and $\wedge$ (cf. [Arieli and Avron, 1998]). Nevertheless, we keep all of the connectives for the ease of presentation. On the other hand, it would be easy to incorporate some additional connectives in the languages $PROP^4_{PS}$ and its restrictions $PROP^3_{PS}$ and $PROP^2_{PS}$. Among them are:

- $\phi \Leftrightarrow \psi =_{def} (\phi \supset \psi) \wedge (\psi \supset \phi)$;
- $\phi \rightarrow \psi =_{def} (\phi \supset \psi) \wedge (\neg\psi \supset \neg\phi)$;
- $\phi \leftrightarrow \psi =_{def} (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$;
- $\phi! =_{def} \phi \wedge \neg\phi$;
- $\Box\phi =_{def} (\neg\phi \supset false) \wedge \neg(\phi \supset \neg\phi)$;
- $\Diamond\phi =_{def} \neg\Box\neg\phi$;

Table 1: Truth tables.

α	β	$\neg\alpha$	$\alpha \wedge \beta$	$\alpha \vee \beta$	$\alpha \supset \beta$	$\alpha \otimes \beta$	$\alpha \oplus \beta$
0	0	1	0	0	1	0	0
0	1	1	0	1	1	$\bot$	$\top$
0	$\top$	1	0	$\top$	1	0	$\top$
0	$\bot$	1	0	$\bot$	1	$\bot$	0
1	0	0	0	1	0	$\bot$	$\top$
1	1	0	1	1	1	1	1
1	$\top$	0	$\top$	1	$\top$	1	$\top$
1	$\bot$	0	$\bot$	1	$\bot$	$\bot$	1
$\top$	0	$\top$	0	$\top$	0	0	$\top$
$\top$	1	$\top$	$\top$	1	1	1	$\top$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\top$	0	1	$\bot$	$\bot$	$\top$
$\bot$	0	$\bot$	0	$\bot$	1	$\bot$	0
$\bot$	1	$\bot$	$\bot$	1	1	$\bot$	1
$\bot$	$\top$	$\bot$	0	1	1	$\bot$	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	1	$\bot$	$\bot$

- $\sim \phi =_{def} \Box\neg\phi$;
- $\odot\phi =_{def} \Box\phi \vee \Box\neg\phi$;
- $\phi \leq \psi =_{def} (\Box\phi \wedge \Box\psi) \vee (\Box\neg\phi \wedge \Box\neg\psi) \vee (\neg\odot\psi \wedge (\psi \vee \neg\psi)) \vee (\neg\odot\phi \wedge ((\phi \wedge \neg\phi) \supset false))$.

These connectives have been taken into account in the languages of some of the logics analyzed in this paper. Thus, the language used in [Arieli and Avron, 1998] to define $FOUR$ is the extension of $PROP^4_{PS}$ where $\rightarrow$ and $\leftrightarrow$ are used as additional (binary) connectives. The language of the logic of 3-inference of [Levesque, 1989] (resp. the language of LP [Priest, 1989] or equivalently the language of RP [Frisch, 1987]) is the propositional language over PS generated from the constant $false$ and the connectives $\neg$, $\wedge$, $\vee$ (resp. from the connectives $\neg$, $\wedge$, $\vee$); $\supset$ and $\Leftrightarrow$ are also introduced but as syntactic sugars ($\phi \supset \psi =_{def} \neg\phi \vee \psi$ and $\phi \Leftrightarrow \psi =_{def} (\phi \supset \psi) \wedge (\psi \supset \phi)$). The language of LP_m [Priest, 1989; Priest, 1991] is the propositional language over PS generated from the connectives $\neg$, $\wedge$, $\vee$ and ! [1]. The language of J_3 is the propositional language over PS generated from the connectives $\neg$, $\wedge$, $\vee$, $\supset$, $\Leftrightarrow$, $\Box$, $\Diamond$, $\sim$, $\odot$. The language considered in [Besnard and Schaub, 1997] is the propositional language over PS generated from the constants $true$, $false$ and the connectives $\neg$, $\wedge$, $\vee$, $\supset$, $\Leftrightarrow$, $\Box$, $\leq$.

[1] To be more precise, ! is introduced as a notation of the metalanguage in [Priest, 1989], and an element of the object language in [Priest, 1991].

The semantics of all such derived connectives can be easily defined by considering their truth tables. For instance, $\Box\phi$ means that ϕ is *necessarily* true, i.e., for every 4-interpretation I over PS, $I(\Box\phi) = 1$ if $I(\phi) = 1$ and $I(\Box\phi) = 0$ otherwise. $\odot\phi$ means that ϕ is classical, i.e., for every 4-interpretation I over PS, $I(\odot\phi) = 1$ if $I(\phi) = 1$ or $I(\phi) = 0$ and $I(\odot\phi) = 0$ otherwise. $\phi \leq \psi$ means that the truth value of ϕ must be lower or equal to the truth value of ψ w.r.t. Belnap's knowledge ordering $\leq_k$ defined by the reflexive-transitive closure of $<_k$ s.t. $\bot <_k 0$, $\bot <_k 1$, $0 <_k \top$ and $1 <_k \top$. The main point here is that there is no way to increase the expressivity of $PROP^4_{PS}$ or $PROP^3_{PS}$ through the incorporation of such additional connectives, since none of them can be genuine. Indeed, just like $PROP^2_{PS}$ is functionally complete for TWO, $PROP^4_{PS}$ is functionally complete for $FOUR$ (Theorem 3.8 from [Arieli and Avron, 1998]) and $PROP^3_{PS}$ is functionally complete for $THREE$ (Theorem 5.1 from [Arieli and Avron, 1998]). This contrasts with the language of J_3 [D'Ottaviano and da Costa, 1970] (and its restrictions like LP [Priest, 1989], RP [Frisch, 1987] and the logic of 3-inference of Levesque [Levesque, 1989]) which does include neither $\oplus$ nor *both*, and despite the presence of other connectives, is not expressive enough to enable the representation of a formula equivalent to *both* (see [Epstein, 1990]). The functional completeness of $PROP^4_{PS}$ (resp. $PROP^3_{PS}$) w.r.t. $FOUR$ (resp. $THREE$) explains why the focus has been laid on them; in some sense, all the logics considered in this paper (including classical logic) are restrictions of $FOUR$.

Such an expressive power of *FOUR* and its restrictions, together with their standard truth functional semantics, is a major feature of these logics. Actually, many usual, intuitive laws of classical logic (including De Morgan's ones, involution of negation, distributivity of $\wedge$ over $\vee$ and vice-versa) still hold in such logics[2].

3 PARACONSISTENT INFERENCE RELATIONS

We are now in position to define several paraconsistent inference relations based on *FOUR*, *THREE* and their restrictions. We start from the simplest ones to the more sophisticated ones.

3.1 THE BASIC RELATIONS

The basic relations are defined in the obvious way from the notions of 4-models and 3-models:

Definition 4 ($\models^4$-inference and $\models^3$-inference) *Let Σ and γ be formulas from $PROP^4_{PS}$ (resp. $PROP^3_{PS}$). We note $\Sigma \models^4 \gamma$ (resp. $\Sigma \models^3 \gamma$) iff every 4-model (resp. 3-model) of Σ is a 4-model (resp. 3-model) of γ.*

Example 2 *Let $\Sigma = a \wedge \neg a \wedge b$. We have $\Sigma \models^4 a$, $\Sigma \models^4 \neg a$ and $\Sigma \models^4 b$, but we do* **not** *have $\Sigma \models^4 \neg b$, nor $\Sigma \models^4 c$. Similar conclusions can be drawn using $\models^3$ instead of $\models^4$. However, both relations do not coincide: we have true $\models^3 a \vee \neg a$ while we do* **not** *have true $\models^4 a \vee \neg a$ (interpreting a as $\bot$, $a \vee \neg a$ is interpreted as $\bot$ as well).*

As evoked before, both J_3, *LP*, *RP* and Levesque's logic of 3-inference can be viewed as restricted cases of *THREE* (what differs is the underlying language which is a subset of $PROP^3_{PS}$):

Definition 5 ($\models_{J_3}$-, $\models_L$- and $\models_{LP}$-inference) *Let Σ and γ be formulas from the language of J_3 (resp. Levesque's logic of 3-inference, LP) as stated in Section 2. We note $\Sigma \models_{J_3} \gamma$ (resp. $\Sigma \models_L \gamma$, $\Sigma \models_{LP} \gamma$) iff every 3-model of Σ is a 3-model of γ.*

A similar definition can be given for $\models_{RP}$. Since $\models_{RP} = \models_{LP}$ holds, we will mainly focus on $\models_{LP}$ in the following.

[2]Note nevertheless that these logics are typically not self-extensional, which implies that only a weak replacement theorem holds (despite the fact that all connectives are truth functional ones). This can be explained by the fact that two truth values are designated. For instance, while $a \vee \neg a \equiv^3 b \vee \neg b$ holds, we do **not** have $\neg(a \vee \neg a) \equiv^3 \neg(b \vee \neg b)$.

Each of the relations $\models^4$, $\models^3$, $\models_{J_3}$, $\models_L$ and $\models_{LP}$ is paraconsistent [Arieli and Avron, 1998], i.e., whenever $\Sigma \in PROP^2_{PS}$ has no 2-model, it does **not necessarily** [3] mean that every formula γ from $PROP^2_{PS}$ satisfies $\Sigma \models^4 \gamma$, $\Sigma \models^3 \gamma$, $\Sigma \models_{J_3} \gamma$, $\Sigma \models_L \gamma$, or $\Sigma \models_{LP} \gamma$. However, they are **too cautious**. For instance, the disjunctive syllogism inference rule of classical logic is not satisfied by any of them. Thus, we do **not** have

$$p \wedge (\neg p \vee q) \models^4 q.$$

This is due to the fact that every 4-interpretation I s.t. $I(p) = \top$ and $I(q) = 0$ is a 4-model of $p \wedge (\neg p \vee q)$, but not a 4-model of q. Especially, none of the relations above coincides with classical entailment $\models^2$ on the $PROP^2_{PS}$ fragment in the situation where the database Σ has a 2-model - while we would expect it.

3.2 THE REFINED RELATIONS

In order to circumvent such difficulties, these inference relations have been refined. The key idea is to restrict the set of models under consideration by taking advantage of some *preference criteria*, so as to keep as many information as possible. This led to the paraconsistent inference relations $\models^4_{I1}$, $\models^4_{I2}$, $\models_{LP_m}$, $\models_{BS}$.

Let us first focus on $\models^4_{I1}$ and $\models^4_{I2}$ [Arieli and Avron, 1998]. Corresponding to them are two preference criteria. The first one consists in giving more credit to the 4-models of Σ that minimize the amount of inconsistent beliefs in Σ. The second one leads to prefer the 4-models of Σ which are as close as possible to its classical models. Formally, two partial preorderings $\leq_1$ and $\leq_2$ over the set of 4-interpretations over *PS* can be used to capture these criteria:

- $I \leq_1 J$ iff $\{x \in PS \mid I(x) \in \{\top\}\} \subseteq \{x \in PS \mid J(x) \in \{\top\}\}$;
- $I \leq_2 J$ iff $\{x \in PS \mid I(x) \in \{\top, \bot\}\} \subseteq \{x \in PS \mid J(x) \in \{\top, \bot\}\}$.

The preferred 4-models of Σ w.r.t. the first (resp. second) preference criterion is the set of 4-models of Σ that are minimal w.r.t. $\leq_1$ (resp. $\leq_2$).

Definition 6 ($\models^4_{I1}$-inference and $\models^4_{I2}$-inference) *Let Σ and γ be formulas from $PROP^4_{PS}$. We note $\Sigma \models^4_{I1} \gamma$ (resp. $\Sigma \models^4_{I2} \gamma$) iff every 4-model of Σ that is minimal in its set w.r.t. $\leq_1$ (resp. $\leq_2$) is a 4-model of γ.*

[3]Trivialization is not avoided in every situation; for instance, the set of (nonstandard) consequences of *false* is the whole representation language, whatever the inference relation among those considered in the paper - as soon as *false* belongs to the language, of course.

Example 3 *Let* $\Sigma = a \wedge \neg a \wedge b \wedge (\neg b \vee c)$. *We have* $\Sigma \models^4_{I1} c$ *and* $\Sigma \models^4_{I2} c$ *(while we do* **not** *have* $\Sigma \models^4 c$*).*

In [Arieli and Avron, 1998], it is shown that both $\models^4_{I1}$ and $\models^4_{I2}$ are valuable inference relations since they are (four-valued) preferential, i.e., any of them satisfies reflexivity, left logical equivalence, right weakening, or, cautious left monotonicity and cautious cut. Obviously, each of them is a proper superset of $\models^4$ but the two relations do not coincide (see counterexamples in [Arieli and Avron, 1998]).

The preference criteria encoded by $\leq_1$ and $\leq_2$ have also been considered in a three-valued framework [Priest, 1989; Priest, 1991] in order to design the logic LP_m, less cautious than LP. Formally, the partial preordering $\leq_{LP_m}$ over the set of 3-interpretations over PS defined by $I \leq_{LP_m} J$ iff $\{x \in PS \mid I(x) \in \{\top\}\} \subseteq \{x \in PS \mid J(x) \in \{\top\}\}$ is considered. Obviously, the sole difference between $\leq_1$ - equal to $\leq_2$ when $\bot$ is not allowed - and $\leq_{LP_m}$ is the underlying set of nonclassical interpretations.

Definition 7 ($\models_{LP_m}$-inference)
Let Σ *and* γ *be formulas from the language of* LP_m. *We note* $\Sigma \models_{LP_m} \gamma$ *iff every 3-model of* Σ *that is minimal in its set w.r.t.* $\leq_{LP_m}$ *is a 3-model of* γ.

Another preference criterion is obtained by giving more importance to the syntax of the database. Indeed, when Σ is a finite *set* of formulas, it is possible to give more credit to the 3-models of Σ that maximize (w.r.t. $\subseteq$) the subset of formulas from Σ that are interpreted to 1. In formal terms, the following partial preordering $\leq_{BS}$ over the set of 3-interpretations over PS is considered: $I \leq_{BS} J$ iff $\{\phi \in \Sigma \mid I(\phi) = 1\} \supseteq \{\phi \in \Sigma \mid J(\phi) = 1\}$.

Definition 8 ($\models_{BS}$-inference)
Let Σ *be a finite set of formulas and* γ *be a formula from the language considered in [Besnard and Schaub, 1997]. We note* $\Sigma \models_{BS} \gamma$ *iff every 3-model of every formula from* Σ *that is minimal in its set w.r.t.* $\leq_{BS}$ *is a 3-model of* γ.

Example 4 *Let* $\Sigma = \{a, \neg a, (a \vee b), (\neg a \vee b)\}$. *We have* $\Sigma \models_{BS} b$ *(while we do* **not** *have* $\Sigma \models_{LP_m} b$ *or* $\Sigma \models^4_{I1} b$ *or* $\Sigma \models^4_{I2} b$*).*

It is easy to observe that both $\models^4_{I1}$, $\models^4_{I2}$, $\models_{LP_m}$ and $\models_{BS}$ are less cautious inference relations than those considered before. Especially, each of them coincides with classical entailment $\models^2$ under the CNF fragment provided that the database Σ has a 2-model and non-tautological queries γ are considered.

To conclude this section, let us also consider some paraconsistent inference relations introduced in [Besnard and Schaub, 1998]. Prima facie, this work is not directly relevant to the multivalued logic approach to inconsistency handling but to the approach based on the selection of consistent subsets. We show nevertheless that it is closely connected to LP_m.

The language considered in [Besnard and Schaub, 1998] is the propositional language over PS generated from the connectives $\neg$, $\wedge$, $\vee$, $\supset$ and $\Leftrightarrow$, where $\supset$ and $\Leftrightarrow$ are classically interpreted. As in [Besnard and Schaub, 1998], we will assume in the following that the database Σ is a *NNF formula* from $PROP^2_{PS}$, i.e., a formula built up from the connectives $\neg$, $\wedge$, $\vee$, only, and for which the scope of every occurrence of $\neg$ is a propositional symbol from PS[4]. Every formula Σ is associated to a default theory $\langle \Sigma^{\pm}, D_{\Sigma} \rangle$ where:

- $\Sigma^{\pm}$ is a formula in the language $PROP^2_{PS^{\pm}}$ where $PS^{\pm} = \{x^+ \mid x \in PS\} \cup \{x^- \mid x \in PS\}$; $\Sigma^{\pm}$ is obtained by replacing in Σ every occurrence of a positive literal x by the positive literal x^+ and every occurrence of a negative literal $\neg x$ by the positive literal x^-.
- $D_{\Sigma} = \{\delta_x \mid x \in PS\}$ is a set of default rules

$$\delta_x = \frac{: x^+ \Leftrightarrow \neg x^-}{(x \Leftrightarrow x^+) \wedge (\neg x \Leftrightarrow x^-)}$$

In this framework, negation is given a special treatment; a first step consists in making every literal independent of its negation through renaming; then the corresponding dependence relations are reintroduced in a parsimonious way, so that inconsistency is avoided. Based on the extensions of the default theory $\langle \Sigma^{\pm}, D_{\Sigma} \rangle$, several paraconsistent consequence relations can be defined. Among them are $\vdash_s$ and $\vdash^{\pm}_s$, defined as follows.

Definition 9 ($\vdash_s$-inference and $\vdash^{\pm}_s$-inference)
Let Σ *and* γ *be two formulas from* $PROP^2_{PS}$.

- γ *is a* skeptical unsigned consequence *of* Σ, *noted* $\Sigma \vdash_s \gamma$, *iff* γ *belongs to every extension of* $\langle \Sigma^{\pm}, D_{\Sigma} \rangle$.
- γ *is a* skeptical signed consequence *of* Σ, *noted* $\Sigma \vdash^{\pm}_s \gamma$, *iff* $\gamma^{\pm}$ *belongs to every extension of* $\langle \Sigma^{\pm}, D_{\Sigma} \rangle$.

Example 5 *Let* $\Sigma = (\neg a \vee b) \wedge a \wedge \neg b \wedge c \wedge (\neg c \vee d)$.

[4]The NNF assumption can be relaxed through the notion of polarity (see [Besnard and Schaub, 1998]) but every occurrence of a subformula $\phi \Leftrightarrow \psi$ must be replaced first by $(\phi \supset \psi) \wedge (\psi \supset \phi)$.

- $\Sigma \vdash_s c \wedge d \wedge (a \vee \neg b)$ *but* $\Sigma \not\vdash_s a$, $\Sigma \not\vdash_s \neg b$ *and* $\Sigma \not\vdash_s (\neg a \vee b)$.
- $\Sigma \vdash_s^{\pm} c \wedge d \wedge a \wedge \neg b \wedge (\neg a \vee b)$.

Both inference relations coincide with classical entailment as soon as Σ has a 2-model. The following proposition shows that $\vdash_s^{\pm}$ actually coincides with $\models_{LP_m}$:

Proposition 1 *Let Σ and γ be formulas from $PROP^2_{PS}$[5]. We have $\Sigma \vdash_s^{\pm} \gamma$ iff $\Sigma \models_{LP_m} \gamma$.*

4 COMPLEXITY RESULTS

In the following, the complexity of the inference problems for the various paraconsistent inference relations considered above is analyzed. We assume that the reader is familiar with some basic notions of computational complexity (see [Papadimitriou, 1994] for details), especially the complexity classes P, NP, coNP and the complexity classes Σ_2^p and Π_2^p of the polynomial hierarchy.

For each inference relation $\vdash$, the input of the decision problem is a pair $\langle \Sigma, \gamma \rangle$ of formulas from the language of the logic under consideration, and the question is "does $\Sigma \vdash \gamma$ holds?". Three cases are successively considered, from the more general one to the more specific one:

- general case: no restriction is put over Σ and γ;
- $\{\neg, \wedge, \vee\}$ fragment: both Σ and γ are required to be generated using these three connectives only;
- Σ CNF and γ propositional symbol. Abusing words, this means that Σ is a finite set of CNF formulas (and γ a symbol) when $\models_{BS}$ is considered.

4.1 MAIN RESULTS

Our main purpose here is to complete the complexity results reported in [Levesque, 1989] and [Cadoli and Schaerf, 1996], which mainly concern the basic relations considered in the paper ($\models^4$, $\models^3$, $\models_{J_3}$, $\models_{LP}$ and $\models_L$). We have obtained the next proposition:

Proposition 2

1. *Let Σ and γ be two formulas from $PROP^4_{PS}$ (resp. $PROP^3_{PS}$, J_3, LP and Levesque's logic) and $\vdash$ be $\models^4$ (resp. $\models^3$, $\models_{J_3}$, $\models_{LP}$ and $\models_L$). Then $\Sigma \stackrel{?}{\vdash} \gamma$ is* coNP*-complete, even if Σ and γ belongs to the $\{\neg, \wedge, \vee\}$ fragment. For Σ in CNF and γ a symbol, $\Sigma \stackrel{?}{\vdash} \gamma$ is in* P.
2. *Let Σ and γ be two formulas from $PROP^4_{PS}$ (resp. $PROP^4_{PS}$, LP_m, the language given in [Besnard and Schaub, 1997], $PROP^2_{PS}$ and $PROP^2_{PS}$) and $\vdash$ be $\models^4_{I1}$ (resp. $\models^4_{I2}$, $\models_{LP_m}$, $\models_{BS}$, $\vdash_s^{\pm}$ and $\vdash_s$). Then $\Sigma \stackrel{?}{\vdash} \gamma$ is Π_2^p-complete, even if Σ is in CNF and γ is a symbol.*

Accordingly, the inference problem for the basic relations considered in the paper is just *as hard* as the inference problem for classical entailment in the general case, and *even easier* (unless P = NP) in the restricted case where Σ is a CNF formula and γ is a propositional symbol. Furthermore, all the complexity results reported in the rightmost column of Table 2 still hold if γ is a CNF formula.

Contrastingly, our results show that the inference problem for the refined relations ($\models^4_{I1}$, $\models^4_{I2}$, $\models_{LP_m}$, $\models_{BS}$) as well as for $\vdash_s^{\pm}$ and $\vdash_s$, is *strictly harder* than the inference problem for classical entailment in the general case (unless the polynomial hierarchy collapses). This is not very surprising due to the preferred models characterization of such relations (intuitively, it is not sufficient to consider all the 4-models or the 3-models of Σ but some computational effort must be spent to select the preferred ones); indeed, a similar computational shift can be observed in classical logic when minimal models are focused on (for various minimality criteria, e.g., those at work in closed world reasoning).

More surprisingly, the Π_2^p-hardness result still holds when Σ is a CNF formula and γ is a propositional symbol, while the corresponding inference problem is in P when no preference over the set of nonclassical interpretations is taken into account. Let us consider $\models^4$ and $\models^4_{I1}$ to give an intuition about it. We know that $\Sigma \models^4 \gamma$ holds iff the formula $\Sigma_{\neg\gamma}$ obtained by replacing in Σ every negated (resp. positive) occurrence of the propositional symbol γ by *true* (resp. *false*) has no 4-model [Cadoli and Schaerf, 1996]; this can be checked in time linear in $|\Sigma_{\neg\gamma}|$ since it is equivalent to check whether $I_{max}(\Sigma_{\neg\gamma}) \notin \{1, \top\}$, where I_{max} is the greatest 4-interpretation over PS w.r.t. $\leq_1$ (i.e., for every symbol $x \in PS$, we have $I_{max}(x) = \top$). The point is that such a characterization result does not hold any longer when the minimal 4-models of Σ w.r.t. $\leq_1$ are selected. On the one hand, there can be exponentially many preferred 4-models, and all of them must be taken into account (one source of com-

[5] Note that $\supset$ and $\Leftrightarrow$ are considered as syntactic sugars here ($\phi \supset \psi =_{def} \neg\phi \vee \psi$ and $\phi \Leftrightarrow \psi =_{def} (\phi \supset \psi) \wedge (\psi \supset \phi)$).

Table 2: Complexity results for paraconsistent inference.

$\Sigma \stackrel{?}{\vdash} \gamma$	general case	$\{\neg, \wedge, \vee\}$ fragment	Σ CNF and γ symbol
$\models^4$	coNP-complete	coNP-complete	in P
$\models^3$	coNP-complete	coNP-complete	in P
$\models_{J_3}$	coNP-complete	coNP-complete	in P
$\models_{LP}$	coNP-complete	coNP-complete	in P
$\models_{L}$	coNP-complete	coNP-complete	in P
$\models^4_{I1}$	Π^p_2-complete	Π^p_2-complete	Π^p_2-complete
$\models^4_{I2}$	Π^p_2-complete	Π^p_2-complete	Π^p_2-complete
$\models_{LP_m}$	Π^p_2-complete	Π^p_2-complete	Π^p_2-complete
$\models_{BS}$	Π^p_2-complete	Π^p_2-complete	Π^p_2-complete
$\vdash^{\pm}_{s}$	Π^p_2-complete	Π^p_2-complete	Π^p_2-complete
$\vdash_{s}$	Π^p_2-complete	Π^p_2-complete	Π^p_2-complete

plexity). On the other hand, determining whether a given 4-model I of Σ is not minimal w.r.t. $\leq_1$ is hard as well, since exponentially many candidates must be taken into account in the worst case to find a certificate (a second source of complexity, independent of the first one).

Additionally, as a direct corollary to the proposition above, we obtained that skeptical inference from a default theory is Π^p_2-hard, even when the default theory $\langle \Sigma, D^{sn}_{\Sigma} = \{\delta^{sn}_x \mid x \in PS\}\rangle$ has a very specific format, namely it is a *supernormal default theory* in which each default is of the form

$$\delta^{sn}_x = \frac{: x^+ \Leftrightarrow \neg x^-}{x^+ \Leftrightarrow \neg x^-}$$

and Σ is a monotone CNF formula (i.e., it contains only positive literals). Accordingly, our complexity results complete known hardness results for skeptical inference from a default theory [Gottlob, 1992; Stillman, 1992].

We also derived the following proposition:

Proposition 3 *The inference problem for $\models^4$, $\models^3$, and $\models_{J_3}$ is* coNP*-complete in the restricted case γ is a symbol.*

Note that if Σ is from the $\{\neg, \wedge, \vee\}$ fragment, the complexity falls down to P [Cadoli and Schaerf, 1996]. coNP-hardness is the price to be paid for the improvement of expressive power achieved by the incorporation of the $\supset$ connective in the language in the presence of *false* (none of $\supset$ or *false* can be derived using the $\wedge$, $\vee$, $\neg$ connectives in any of the multivalued logics considered in this paper) and the possibility $\supset$ offers to draw some nontrivial inferences in *FOUR* and *THREE* (unlike disjunctive syllogism, the well-known *modus ponens* rule holds in these logics).

4.2 IMPACT ON OTHER RELATIONS

Interestingly, our complexity results can be easily extended to other three-valued or four-valued paraconsistent inference relations, closely related to those analyzed in the paper. Indeed, let us consider the following basic inference relations (Σ and γ are formulas from $PROP^4_{PS}$):

- $\Sigma \models^4_{\supset} \gamma$ iff $\models^4 \Sigma \supset \gamma$;
- $\Sigma \models^4_{inc.} \gamma$ iff $\Sigma \wedge \neg\gamma$ has no 4-model;
- $\Sigma \models^4_{\leq_t} \gamma$ iff for every 4-interpretation I over PS, we have $I(\Sigma) \leq_t I(\gamma)$, where $\leq_t$ is Belnap's truth ordering defined by the reflexive-transitive closure of $<_t$ s.t. $0 <_t \bot$, $0 <_t \top$, $\bot <_t 1$ and $\top <_t 1$;
- $\Sigma \models^4_{true} \gamma$ iff for every 4-model I of Σ, we have $I(\gamma) = 1$.

All these relations correspond to different ways of defining a notion of consequence within a multivalued logic, and they coincide in classical logic.

It is easy to show that $\models^4_{\supset} = \models^4$. In the general case, we have $\models^4_{true} \subset \models^4_{\leq_t} \subset \models^4$. We also have $\models^4_{true} \subset \models^4_{inc.}$. $\models^4_{inc.}$ and $\models^4_{\leq_t}$ (resp. $\models^4$) are incomparable w.r.t. $\subseteq$ in the general case.

We can also define the corresponding three-valued inference relations (just by replacing every 4 by a 3 above). In this situation, $\models^3_{\supset} = \models^3$ and $\models^3_{inc.} = \models^3_{true}$, and we also have $\models^3_{true} \subset \models^3_{\leq_t} \subset \models^3$ in the general case. Taking advantage of our results, we can prove that the decision problems associated to all these relations are coNP-complete, even in the restricted case γ is a propositional symbol.

Based on similar ideas, some refined relations can be defined. Let $*$ be among 1, 2; we let:

- $\Sigma \models^4_{I_*,\leq_t} \gamma$ iff for every 4-model I of Σ that is minimal w.r.t. $\leq_*$ in its set, we have $I(\Sigma) \leq_t I(\gamma)$.
- $\Sigma \models^4_{I_*,true} \gamma$ iff for every 4-model I of Σ that is minimal w.r.t. $\leq_*$ in its set, we have $I(\gamma) = 1$.

Such refinements lead to more and more cautious relations since in the general case, we have $\models^4_{I_*,true} \subset \models^4_{I_*,\leq_t} \subset \models^4_{I_*}$. Again, the corresponding three-valued inference relations can be obtained by replacing every 4 by a 3 just above (in this case, there is no distinction between $* = 1$ and $* = 2$) and the inclusions above still hold in this case. Unsurprisingly, we can prove that the decision problems associated to all these refined relations are Π^p_2-complete, even in the restricted case Σ is a CNF formula and γ a propositional symbol.

Other paraconsistent inference relations can be tentatively defined by increasing the number of truth values; we then enter the framework of *logical bilattices*, i.e., bilattices on which the sets of designated truth values are prime bifilters (see [Ginsberg, 1988], [Arieli and Avron, 1998]). It has been shown that such an extension of the number of truth values does not add much. Indeed, the corresponding basic inference relations coincide with $\models^4$ (Theorem 6.17 from [Arieli and Avron, 1998]), so our complexity results concerning $\models^4$ immediately apply to them. Furthermore, while extensions to $\models^4_{I1}$ and $\models^4_{I2}$ can be envisioned by preferring models minimizing the set of propositional symbols assigned to an "inconsistent truth value", i.e., those belonging to a given inconsistency set I (see [Arieli and Avron, 1998] for details), it has been shown that this does not lead to inference relations that differ from $\models^4_{I1}$ or $\models^4_{I2}$ (Theorem 6.25 from [Arieli and Avron, 1998]). Since it is possible to determine which one of $\models^4_{I1}$ or $\models^4_{I2}$ is reached independently from the input of the decision problem (Σ and γ), i.e., just by comparing the inconsistency set I with the set of truth values v of the bilattice s.t. neither v nor $\neg v$ belongs to the prime bifilter under consideration, our Π^p_2-completeness results concerning $\models^4_{I1}$ and $\models^4_{I2}$ still apply to such refined relations.

Finally, it is also possible to take advantage of our results to identify the computational complexity of some inference relations proposed so far in the framework of disjunctive logic programming. A basic motivation is to avoid trivial inference from logic programs that do not have any model under usual semantics, like $\mathcal{LP} = \{a \vee b \leftarrow, \neg a \leftarrow, \neg b \leftarrow, c \leftarrow\}$. In a nutshell, the paraconsistent models of an extended disjunctive logic program $\mathcal{LP}$ as defined in [Sakama and Inoue, 1995] are exactly the 4-models of $\mathcal{LP}$ in which every occurrence of $\leftarrow$ is replaced by $\subset$, and every occurrence of *not* l (where *not* is negation as failure, distinct from $\neg$, the "true" negation) is replaced by $(l \supset false)$; this result can be easily established by structural induction on $\mathcal{LP}$. Our complexity results can be extended to show that the corresponding inference problem is coNP-complete, even in the restricted case the query γ is a propositional symbol. Exploiting Proposition 4.14 from [Arieli and Avron, 1998], coNP-completeness still holds when the minimal paraconsistent models of $\mathcal{LP}$ are selected (see [Sakama and Inoue, 1995] for details) and the query belongs to the $\{\wedge, \vee, \neg\}$ fragment. Contrastingly, the complexity of more interesting inference relations based on paraconsistent stable models cannot be identified directly from our results.

5 CONCLUSION

In this paper, the complexity of several paraconsistent inference relations based on multivalued logics has been investigated. A number of inference relations pointed out by Arieli and Avron, D'Ottaviano and da Costa, Priest, Besnard and Schaub, Levesque have been considered from the computational side, in the general case and when some restrictions are put on the database and the query. Our main contribution has been to show that the refined paraconsistent inference relations that have been defined in the framework of multivalued logics are at the second level of the polynomial hierarchy, even in some quite restricted cases. This is where the complexity of many inconsistent-tolerant inference relations proposed so far in the literature lies (especially, those based on the selection of maximally consistent subsets of the database) as well as many other forms of inference considered by the AI community (see e.g., [Cadoli and Schaerf, 1993]). Another contribution has been to show close relationships between some of the relations considered in the paper. Specifically, we proved that $\models_{LP_m}$ coincides with $\vdash^{\pm}_s$ on their common language. On the one hand, this gives a multivalued semantics to $\vdash^{\pm}_s$. On the other hand, this equivalence gives a default logic characterization to LP_m inference. This is particularly helpful from the practical side since the default theory $(\Sigma^{\pm}, D_\Sigma)$ can be replaced by the supernormal default theory $\langle \Sigma^{\pm}, D^{sn}_\Sigma \rangle$, without questioning the set of (skeptical) signed consequences (see Theorems A.1 and A.2 from [Besnard and Schaub, 1998] for details). Furthermore, several implementations of this simple fragment of default logic – like Theorist [Poole, 1988] – exist. Finally, we have also presented some complexity results for related inference relations, both basic and refined

ones.

There are at least two ways to interpret the complexity results we gave. A pessimistic interpretation is that none of the inference relations considered in the paper is really satisfying, since either it is very cautious or it is highly intractable. A more optimistic interpretation is reached when intractability is viewed as a feature and not as a drawback. Observing that $PROP^4_{PS}$ and $PROP^3_{PS}$ are strictly more expressive than $PROP^2_{PS}$ and the CNF fragment of it, our analysis shows that, without any extra computational cost, we can take advantage of quite expressive languages (like full $PROP^4_{PS}$ instead of the language of LP) to encode the database.

This work calls for several perspectives. One of them consists in investigating restrictions on the database that would lead to more tractable inference relations. Especially, it would be interesting to consider the case Σ is a Horn CNF formula (since the satisfiability problem for such formulas is tractable in classical logic). Another way to circumvent the intractability would be to restrict the number of preferred symbols (or formulas) when interpreted classically; a preliminary work relevant to such a resource-bounded approach to paraconsistent inference is [Marquis and Porquet, 2001].

Acknowledgements

Many thanks to the anonymous reviewers and to Jérôme Lang for their comments and suggestions.

The authors have been partly supported by the IUT de Lens, the Université d'Artois, the Région Nord/Pas-de-Calais under the TACT-TIC project, and by the European Community FEDER Program.

References

[Arieli and Avron, 1998] O. Arieli and A. Avron. The value of four values. *Artificial Intelligence*, 102(1):97–141, 1998.

[Baral *et al.*, 1991] C. Baral, S. Kraus, and J. Minker. Combining multiple knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 3(2):208–220, 1991.

[Belnap, 1977] N. Belnap. *Modern Uses of Multiple-Valued Logic*, chapter A useful four-valued logic, pages 8–37. Reidel, 1977.

[Benferhat *et al.*, 1993] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, and H. Prade. Inconsistency management and prioritized syntax-based entailment. In *Proceedings of the* 13^{th} *International Joint Conference on Artificial Intelligence (IJCAI'93)*, pages 640–645, Chambéry (France), 1993.

[Besnard and Hunter, 1995] Ph. Besnard and A. Hunter. Quasi-classical logic: non-trivializable classical reasoning from inconsistent information. In *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU'95)*, volume 946 of *LNAI*, pages 44–51. Springer-Verlag, 1995.

[Besnard and Hunter, 1998] Ph. Besnard and A. Hunter. *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 2, chapter Introduction to actual and potential contradictions, pages 1–11. Kluwer Academic, 1998.

[Besnard and Schaub, 1997] Ph. Besnard and T. Schaub. Circumscribing inconsistency. In *Proceedings of the* 15^{th} *International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 150–155, Nagoya (Japan), 1997.

[Besnard and Schaub, 1998] Ph. Besnard and T. Schaub. Signed systems for paraconsistent reasoning. *Journal of Automated Reasoning*, 20:191–213, 1998.

[Bondarenko *et al.*, 1997] A. Bondarenko, P. Dung, R.A. Kowalski, and F. Toni. An abstract, argumentation-theoretic framework for default reasoning. *Artificial Intelligence*, 93:63–101, 1997.

[Brewka, 1989] G. Brewka. Preferred subtheories: an extended logical framework for default reasoning. In *Proceedings of the* 11^{th} *International Joint Conference on Artificial Intelligence (IJCAI'89)*, pages 1043–1048, Detroit (MI), 1989.

[Cadoli and Schaerf, 1993] M. Cadoli and M. Schaerf. A survey on complexity results for non-monotonic logics. *Journal of Logic Programming*, 17:127–160, 1993.

[Cadoli and Schaerf, 1996] M. Cadoli and M. Schaerf. On the complexity of entailment in propositional multivalued logics. *Annals of Mathematics and Artificial Intelligence*, 18:29–50, 1996.

[da Costa, 1974] N. C. A. da Costa. On the theory of inconsistent formal systems. *Notre Dame Journal of Formal Logic*, 15:497–510, 1974.

[D'Ottaviano and da Costa, 1970] I.M.L. D'Ottaviano and N.C.A. da Costa. Sur un problème de Jaśkowski. In *Comptes Rendus de l'Académie des Sciences de Paris*, volume 270, pages 1349–1353. 1970.

[Dung, 1995] P. Dung. On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

[Eiter and Gottlob, 1992] Th. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence*, 57:227–270, 1992.

[Eiter *et al.*, 1998] T. Eiter, N. Leone, and D. Saccà. Expressive Power and Complexity of Partial Models for Disjunctive Deductive Databases. *Theoretical Computer Science*, 206(1–2):181–218, 1998.

[Elvang-Goransson and Hunter, 1995] M. Elvang-Goransson and A. Hunter. Argumentative logics: reasoning from classically inconsistent information. *Data and Knowledge Engineering*, 16:125–145, 1995.

[Epstein, 1990] R.L. Epstein. *Propositional Logics*, volume 1, chapter The Semantic Foundations of Logic. Kluwer Academic, 1990.

[Fagin *et al.*, 1983] R. Fagin, J.D. Ullman, and M.Y. Vardi. On the semantics of updates in databases. In *Proceedings of the 2^{nd} ACM Symposium on Principles of Database Systems (PODS'83)*, pages 352–355, 1983.

[Frisch, 1987] A.M. Frisch. Inference without chaining. In *Proceedings of the 10^{th} International Joint Conference on Artificial Intelligence (IJCAI'87)*, pages 515–519, Milan (Italy), 1987.

[Ginsberg, 1986] M.L. Ginsberg. Counterfactuals. *Artificial Intelligence*, 30:35–79, 1986.

[Ginsberg, 1988] M.L. Ginsberg. Multivalued logics: a uniform approach to inference in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.

[Gottlob, 1992] G. Gottlob. Complexity results for nonmonotonic logics. *Journal of Logic and Computation*, 2:397–425, 1992.

[Hunter, 1998] A. Hunter. *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 2, chapter Paraconsistent logics, pages 11–36. Kluwer Academic, 1998.

[Hunter, 2000] A. Hunter. Reasoning with contradictory information using quasi-classical logic. *Journal of Logic and Computation*, 10(5):677–703, 2000.

[Konieczny and Pino Pérez, 1998] S. Konieczny and R. Pino Pérez. On the logic of merging. In *Proceedings of the 6^{th} International Conference on Knowledge Representation and Reasoning (KR'98)*, pages 488–498, Trento (Italy), 1998.

[Konieczny, 2000] S. Konieczny. On the difference between merging knowledge bases and combining them. In *Proceedings of the 7^{th} International Conference on Knowledge Representation and Reasoning (KR'00)*, pages 135–144, Breckenridge (CO), 2000.

[Levesque, 1989] H.J. Levesque. A knowledge-level account of abduction (preliminary version). In *Proceedings of the 11^{th} International Joint Conference on Artificial Intelligence (IJCAI'89)*, pages 1061–1067, Detroit (MI), 1989.

[Lin, 1996] J. Lin. Integration of weighted knowledge bases. *Artificial Intelligence*, 83(2):363–378, 1996.

[Marquis and Porquet, 2001] P. Marquis and N. Porquet. Resource-bounded inference from inconsistent belief bases. In *Proceedings of the 17^{th} International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 103–108, Seattle (WA), 2001.

[Nebel, 1992] B. Nebel. *Belief Revision*, chapter Syntax-based approaches to belief revision, pages 52–88. Number 29 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1992.

[Nebel, 1994] B. Nebel. Base revision operations and schemes: semantics, representation and complexity. In *Proc. of the 11^{th} European Conference on Artificial Intelligence (ECAI'94)*, pages 341–345, Amsterdam (Netherlands), 1994.

[Papadimitriou, 1994] Ch. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Pinkas and Loui, 1992] G. Pinkas and R.P. Loui. Reasoning from inconsistency: a taxonomy of principles for resolving conflict. In *Proceedings of the 3^{rd} International Conference on Knowledge Representation and Reasoning (KR'92)*, pages 709–719, Cambridge (MA), 1992.

[Poole, 1988] D. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47, 1988.

[Priest, 1989] G. Priest. Reasoning about truth. *Artificial Intelligence*, 39:231–244, 1989.

[Priest, 1991] G. Priest. Minimally inconsistent LP. *Studia Logica*, 50:321–331, 1991.

[Rescher and Manor, 1970] N. Rescher and R. Manor. On inference from inconsistent premises. *Theory and Decision*, 1:179–219, 1970.

[Revesz, 1997] P.Z. Revesz. On the semantics of arbitration. *International Journal of Algebra and Computation*, 7(2):133–160, 1997.

[Sakama and Inoue, 1995] C. Sakama and K. Inoue. Paraconsistent stable semantics for extended disjunctive programs. *Journal of Logic and Computation*, 5:265–285, 1995.

[Stillman, 1992] J. Stillman. The complexity of propositional default logics. In *Proc. of the 10th National Conference on Artificial Intelligence (AAAI'92)*, pages 794–799, San Jose (CA), 1992.

[Winslett, 1990] M. Winslett. *Cambridge Tracts in Theoretical Computer Science*, chapter Updating logical databases. Cambridge University Press, 1990.

Appendix: proof sketches

Proposition 1: First, we exploit the fact that $\Sigma \vdash_s^{\pm} \gamma$ iff $\gamma^{\pm}$ belongs to every extension of the supernormal default theory $T = \langle \Sigma^{\pm}, \{\frac{: x^+ \Leftrightarrow \neg x^-}{x^+ \Leftrightarrow \neg x^-} \mid x \in PS\}\rangle$ (see Theorems A.1 and A.2 from [Besnard and Schaub, 1998]). Then we take advantage of the fact that $\Sigma^{\pm}$ is a monotone (positive) formula to show that every preferred model I of T is s.t. for every $x \in PS$, we do **not** have $I(x^+) = I(x^-) = 0$. Accordingly, we can associate to each such preferred model I of T over $PS^{\pm}$ in a bijective way the 3-model $3(I)$ of Σ over PS s.t. $3(I)(x) = I(x^+)$ and $3(I)(\neg x) = I(x^-)$. It remains to show that 3() is a morphism (i.e., the preference ordering between interpretations is preserved) to conclude the proof. ■

Proposition 2:

- Membership: The membership-to-P results are easy consequences from [Cadoli and Schaerf, 1996]. As to $\vdash_s$ and $\vdash_s^{\pm}$, the membership to Π_2^p comes directly from Theorem 5.2 from [Gottlob, 1992] (see also [Stillman, 1992]). The membership-to-coNP results are not very difficult: just consider the complementary problems and solve them by guessing an interpretation and verifying that it is a model of Σ but not a model of γ. A similar approach can be followed for the remaining results; the unique difference is that the guessed interpretation must be shown preferred using a call to an NP oracle; this is easy to do by considering the complementary problem (if an interpretation is not preferred then another interpretation can be guessed and checked as strictly more preferred than the first one in deterministic polynomial time).
- Hardness: coNP-hardness results are easy consequences from [Cadoli and Schaerf, 1996]. For Π_2^p-hardness results, we exploit the fact that both $\models_{I1}^4$, $\models_{I2}^4$ and $\vdash_s^{\pm}$ coincide with $\models_{LP_m}$ whenever Σ is a CNF formula and γ a symbol. We proceed through a number of lemmata to prove that determining whether $\Sigma \not\models_{LP_m} \gamma$ is Σ_2^p-hard whenever Σ is a CNF formula and γ is a symbol. The key ones are:
  1. Given a finite set Σ of clauses from $PROP_{PS}^2$, we show that determining whether $x \in Var(\Sigma)$[6] belongs to at least one var-conflict of Σ is Σ_2^p-hard. A var-conflict is a minimal (w.r.t. $\subseteq$) set of variables occurring in a conflict of Σ (i.e., a minimal (w.r.t. $\subseteq$) subset of Σ that has no 2-model). To achieve this Σ_2^p-hardness proof, we reduce the complementary of WIDTIO inference problem [Winslett, 1990] to the latter problem, taking advantage of Lemma 6.2 and Theorem 8.2 from [Eiter and Gottlob, 1992].
  2. Then, we show that $x \in Var(\Sigma)$ belongs to at least one var-conflict of Σ iff Σ has a 3-model I minimal w.r.t. $\leq_{LP_m}$ s.t. $I(x) = \top$.

  Finally, we reduce the inference problem in LP_m to the one in the approach of Besnard and Schaub [Besnard and Schaub, 1997] and to the skeptical unsigned inference problem [Besnard and Schaub, 1998].

■

Proposition 3: The membership results follow from the previous proposition. Hardness is obtained by reduction from UNSAT, the problem of determining whether a CNF formula from $PROP_{PS}^2$ has no 2-model, and is achieved by taking advantage of the $\supset$ connective.

[6] $Var(\Sigma)$ is the subset of propositional symbols occurring in Σ.

Evaluation-Based Reasoning with Disjunctive Information in First-Order Knowledge Bases

Gerhard Lakemeyer
Dept. of Computer Science
RWTH Aachen
52056 Aachen
Germany
gerhard@cs.rwth-aachen.de

Hector J. Levesque
Dept. of Computer Science
University of Toronto
Toronto, Ontario
Canada M5S 3A6
hector@cs.toronto.edu

Abstract

In previous work, Levesque proposed an evaluation-based reasoning procedure for so-called *proper* KBs, equivalent to a possibly incomplete possibly infinite set of function-free ground literals. The procedure, called V, preserved the efficiency and logical soundness of database query evaluation. Moreover, if the query was constrained to be in a special normal form, V was also logically complete. In this paper, we propose an extension to this work to handle disjunctive information in a KB. We define a query evaluation procedure X that generalizes V to deal with KBs that are equivalent to a possibly infinite set of function-free ground *clauses*. Deductive reasoning that is logically sound and complete in this case is undecidable in general, and so, we describe what we are willing to give up in X to preserve tractability.

1 Introduction

It was argued in [8] that the only procedure yet proposed to perform deductive reasoning effectively over extremely large first-order knowledge bases (KBs) (involving say 10^5 or more facts) was *query evaluation* over databases. A database, however, is a very special sort of KB, namely one that is equivalent (under certain assumptions) to a maximally consistent set of function-free ground literals. No incomplete knowledge is allowed. In [8], an evaluation-based reasoning procedure was proposed for proper KBs, equivalent to a possibly *incomplete* set of function-free ground literals. The procedure, called V, preserved the efficiency and logical soundness of database query evaluation. Moreover, if the query was constrained to be in a special normal form, V was also logically complete.

However, these results do not allow any form of disjunctive knowledge in the KB. There can be a very large table for the predicate *GradStudent*, for instance, including both positive and negative instances, and even leaving the predicate open for certain individuals. But the KB cannot include anything like

$$(\mathit{GradStudent}(\mathit{john}) \vee \mathit{GradStudent}(\mathit{mary}))$$

or even

$$\forall x(\mathit{GradStudent}(x) \supset \mathit{Student}(x))$$

as a simple way of duplicating the table for the *Student* predicate. In this paper, we propose an approach to handling disjunctive information in a KB. We define a query evaluation procedure X that generalizes V to deal with KBs that are equivalent to a possibly infinite set of function-free ground *clauses*. Deductive reasoning that is logically sound and complete in this case is undecidable in general, and so, we describe what we are willing to give up in X to preserve tractability.

The rest of the paper is organized as follows. In the next section, we review the evaluation-based reasoning procedure V, and prove a new result, which is that in the case of proper KBs, V actually computes a well-known limited form of reasoning called *tautological entailment*. In Section 3, we discuss the difficulty of generalizing V to KBs with disjunction, and argue that tautological entailment does too little in some ways and too much in others. In Section 4, we define X, an evaluation-based reasoning procedure that deals with disjunction in a novel, more appropriate way. In Section 5, we show that X agrees with V and tautological entailment on proper KBs, that it performs logically sound reasoning, and that it is computable. Finally in Section 6, we discuss future work.

2 Proper Knowledge Bases

We start with a standard first-order language $\mathcal{L}$ with no function symbols other than constants and a distinguished

equality predicate. We assume a countably infinite set of constants $\mathcal{C} = \{c_1, c_2, \ldots\}$ for which we will be making a unique-name assumption.

Notation: As usual, elements of $\mathcal{L}$ are called formulas and formulas without free variables are called sentences. We only consider the logical connectives $\neg$, $\vee$, and $\exists$ as part of the language, but we will freely use $\wedge$, $\forall$, and $\supset$ as the usual abbreviations. We will let l range over literals and we will write $\bar{l}$ to denote its complement. Clauses are, as usual, disjunctions of literals and we will sometimes write them using set notation. We will use θ to range over substitutions of all variables by constants, and write $\alpha\theta$ as the result of applying the substitutions to α. We will write α^x_d to denote α with all free occurrences of x replaced by d. We will let ρ range over atoms whose arguments are distinct variables, so that $\rho\theta$ ranges over ground atoms.[1] We will use $\forall\alpha$ to mean the universal closure of α. Finally, we will use e to range over *ewffs*, by which we mean quantifier-free formulas whose only predicate is equality.

V and later also X make the assumption that quantification can be understood substitutionally with respect to $\mathcal{C}$. This assumption corresponds to restricting one's attention to the following kind of interpretations:

Definition 1: A *standard* interpretation of $\mathcal{L}$ is one where $=$ is interpreted as identity, and the denotation relation between $\mathcal{C}$ and the domain of discourse is bijective.

As the following definition and theorem from [8] show, this restriction can be captured precisely using a set of axioms about equality, provided we consider logical theories which do not mention infinitely many constants, which is something finite knowledge bases always satisfy.[2]

Definition 2: The set $\mathcal{E}$ is the axioms of equality (equivalence relation, substitution of equals for equals) and the (infinite) set of formulas $\{(c_i \neq c_j) \mid i \neq j\}$.

Theorem 1: *(from [8])*
Suppose S is any set of closed formulas, and that that there is an infinite set of constants that do not appear in S. Then $\mathcal{E} \cup S$ is satisfiable iff it has a standard model.

Knowledge bases considered by V have the following form:

Definition 3: A set KB of formulas is called *proper* if $\mathcal{E} \cup KB$ is consistent and KB is a finite set of formulas of the form $\forall(e \supset \rho)$ or $\forall(e \supset \neg\rho)$.

[1] Because equality is treated separately, "atoms" and "literals" here do not include equalities.

[2] See also [9] where logics are considered based only on standard interpretations.

Note that restricting ρ to be an atom with only variables as arguments is no restriction. For example, if d is a constant then $\neg P(d)$ can be rewritten as $\forall x.(x = d) \supset \neg P(x)$.

Given a proper knowledge base KB and a query, V returns one of three values 0, 1, or $\frac{1}{2}$, where 0 means "known to be false," 1 means "known to be true," and $\frac{1}{2}$ means "unknown." Then we have

1. $V[\rho\theta] = \begin{cases} 1 & \text{if there is a } \forall(e \supset \rho) \in KB \\ & \text{such that } V[e\theta] = 1 \\ 0 & \text{if there is a } \forall(e \supset \neg\rho) \in KB \\ & \text{such that } V[e\theta] = 1 \\ \frac{1}{2} & \text{otherwise} \end{cases}$
2. $V[t = t'] = 1$ if t is identical to t', and 0 otherwise.
3. $V[\neg\alpha] = 1 - V[\alpha]$.
4. $V[\alpha \vee \beta] = \max\{V[\alpha], V[\beta]\}$.
5. $V[\exists x.\alpha] = \max_{d \in H_1^+} \{V[\alpha^x_d]\}$

Here, H_k^+ is the union of the constants in KB, those mentioned in the query α, and k new constants appearing nowhere in KB and α.

The notions of soundness and completeness for evaluation methods like V was introduced in [8] as follows:

Definition 4: Let $S, T \subseteq \mathcal{L}$, and let $f \in [\mathcal{L} \to \{0, 1, \frac{1}{2}\}]$. Then

- f is logically *sound* wrt S for T iff for every $\alpha \in T$, if $f[\alpha] = 1$ then $S \models \alpha$, and if $f[\alpha] = 0$ then $S \models \neg\alpha$;
- f is logically *complete* wrt S for T iff for every $\alpha \in T$, if $S \models \alpha$ then $f[\alpha] = 1$, and if $S \models \neg\alpha$ then $f[\alpha] = 0$.

The proof of the soundness of V relies on the following theorem, which was proved in [8]:

Theorem 2: *(from [8])*
Let S be a set of closed formulas, let α be a formula with a single free variable x. Then for every constant $d \in \mathcal{C}$, there is a constant $c \in H_1^+(S \cup \{\alpha\})$ such that $\mathcal{E} \cup S \models \alpha^x_d$ iff $\mathcal{E} \cup S \models \alpha^x_c$.

Since V is decidable, it clearly is not complete for arbitrary queries. However, Levesque showed completeness for queries in a certain normal form called $\mathcal{NF}$. In the propositional case, for example, conjunctions of the non-tautologous prime implicates of a sentence are in $\mathcal{NF}$. In the first-order case, the situation is more complex. For the purposes of this paper, it suffices to note that any negation-free sentence is in $\mathcal{NF}$.

Interestingly, when we replace logical entailment by a variant of *tautological entailment*, a fragment of relevance logic [2], we obtain that V is complete for *arbitrary* queries. To show this, we first define tautological entailment semantically.

For simplicity, we define the semantics of tautological entailment only for standard models and, in addition, keep the semantics of equality classical. (It can easily be shown that our variant of tautological entailment coincides with the original one for sentences without equality.)

Let us call a set s of ground literals (not necessarily consistent) a *setup*. Then the support relation $\models^t$ between setups and sentences is defined as follows:

1. $s \models^t l$ iff $l \in s$ for any literal l;
2. $s \models^t (t = t')$ iff t is identical to t';
3. $s \models^t \neg(t_1 = t_2)$ iff t is not identical to t';
4. $s \models^t \neg\neg\alpha$ iff $s \models^t \alpha$;
5. $s \models^t (\alpha \vee \beta)$ iff $s \models^t \alpha$ or $s \models^t \beta$;
6. $s \models^t \neg(\alpha \vee \beta)$ iff $s \models^t \neg\alpha$ and $s \models^t \neg\beta$;
7. $s \models^t \exists x.\alpha$ iff $s \models^t \alpha^x_d$ for some $d \in \mathcal{C}$;
8. $s \models^t \neg\exists x.\alpha$ iff $s \models^t \neg\alpha^x_d$ for all $d \in \mathcal{C}$;

Note that because setups are arbitrary sets of literals and because there are separate rules of interpretation for sentences and their negations, the truth and falsity of a sentence may receive independent support.[3]

Definition 5: A set of sentences S *tautologically entails* a sentence α ($S \longrightarrow \alpha$) iff for all setups s, if $s \models^t \gamma$ for all $\gamma \in S$, then $s \models^t \alpha$.

By breaking the connection between the truth and the falsity of a sentence, tautological entailment becomes strictly weaker than logical entailment. For example, $p \wedge (p \supset q)$ does not tautologically entail q because there are setups which support both p and $\neg p$ and, hence, $p \wedge (p \supset q)$, yet do not support q. It turns out that this weakening is just enough to make V complete for tautological entailment and arbitrary queries.

To prove the theorem we need the following definition and lemmas.

Definition 6: For a proper *KB*, let
Lits(KB) $= \{l\theta \mid \forall(e \supset l) \in KB$ and $\mathcal{E} \models e\theta\}$.

It is easy to see that any setup which satisfies *KB* is a superset of *Lits(KB)*. In other words, *Lits(KB)* is the minimal setup which satisfies *KB*.

[3]Originally, setups were defined using the four truth values *true*, *false*, *neither*, and *both* assigned to atoms [4]. Defining them, as we do, in terms of sets of literals is equivalent.

Lemma 3: *For every setup s such that $s \models^t KB$, if Lits(KB) $\models^t \alpha$ then $s \models^t \alpha$*

Proof: The proof is by a simple induction on the structure of α using the fact that any setup which satisfies *KB* is a superset of *Lits(KB)*. ∎

Lemma 4: *$KB \longrightarrow \alpha$ iff Lits(KB) $\models^t \alpha$.*

Proof: Let $KB \longrightarrow \alpha$. Since *Lits(KB)* $\models^t KB$, *Lits(KB)* $\models^t \alpha$ follows. Conversely, let *Lits(KB)* $\models^t \alpha$. Then by Lemma 3 $s \models^t \alpha$ for any s such that $s \models^t KB$. ∎

Lemma 5: *$KB \longrightarrow \exists x.\alpha$ iff $KB \longrightarrow \alpha^x_c$ for some $c \in H_1^+$.*

The proof is almost identical to that of of Theorem 2, which can be found in [8]. See also the proof of Theorem 14 in this paper, where a similar argument is used.

Theorem 6: *Let KB be proper and α an arbitrary sentence. Then*
$V[\alpha] = 1$ iff $KB \longrightarrow \alpha$ and $V[\alpha] = 0$ iff $KB \longrightarrow \neg\alpha$.

Proof: The proof is by induction on the structure of α.

For closed atomic formulas $\rho\theta$ we have $V[\rho\theta] = 1$ iff there is a $\forall(e \supset \rho) \in KB$ such that $V[e\theta] = 1$ iff $\rho\theta \in$ *Lits(KB)* (since$V[e\theta] = 1$ iff $\mathcal{E} \models e\theta$ by Lemma 7 of [8]) iff *Lits(KB)* $\models^t \rho\theta$ iff $KB \longrightarrow \rho\theta$ by Lemma 4. The case for $V[\rho\theta] = 0$ is similar.

$V[t = t'] = 1$ iff t and t' are identical iff $s \models^t (t = t')$ for all setups s iff $KB \longrightarrow (t = t')$, and similarly for the case $V[t = t'] = 0$.

$V[\neg\alpha] = 1$ iff $V[\alpha] = 0$ iff (by induction) $KB \longrightarrow \neg\alpha$, and similarly for $V[\neg\alpha] = 0$.

$V[\alpha \vee \beta] = 1$ iff $V[\alpha] = 1$ or $V[\beta] = 1$ iff (by induction and Lemma 4) *Lits(KB)* $\models^t \alpha$ or *Lits(KB)* $\models^t \beta$ iff *Lits(KB)* $\models^t (\alpha \vee \beta)$ iff $KB \longrightarrow (\alpha \vee \beta)$, again by Lemma 4. The case $V[\alpha \vee \beta] = 0$ is proved analogously.

$V[\exists x.\alpha] = 1$ iff $V[\alpha^x_c] = 1$ for some $c \in H_1^+$ iff (by induction) $KB \longrightarrow \alpha^x_c$ iff (Lemma 4) *Lits(KB)* $\models^t \alpha^x_c$ iff *Lits(KB)* $\models^t \exists x.\alpha$ (by Lemma 5) iff $KB \longrightarrow \exists x.\alpha$. Similarly for $V[\exists x.\alpha] = 0$. ∎

This theorem shows that for proper KBs, V coincides with tautological entailment. It therefore follows that for proper KBs, tautological entailment is decidable and moreover agrees with logical entailment on queries in $\mathcal{NF}$.

3 Including clauses in a KB

In the previous section, we saw that it was possible to efficiently perform logically sound first-order reasoning us-

ing an evaluation-based procedure if the KB in question was proper. Furthermore, the reasoning was logically complete if the query was in a normal form, $\mathcal{NF}$. But a proper KB is one that is equivalent to a (possibly infinite) set of ground *literals*. What we now want to consider is this: is there an evaluation-based reasoning procedure that works for KBs that are equivalent to a (possibly infinite) set of ground *clauses*? We will argue that the answer is *yes*.

3.1 Negation is not the problem

Of course, the entailment problem we are contemplating will be undecidable in general, just as it was for proper KBs. However, unlike with proper KBs, the problem remains undecidable even for queries in $\mathcal{NF}$:

Theorem 7: *The question as to whether KB $\models \alpha$ is undecidable, even if α is negation-free and KB is of the form* $\{\forall(P_1(\vec{x}_1) \vee Q_1(\vec{x}_1)), \ldots, \forall(P_n(\vec{x}_n) \vee Q_n(\vec{x}_n))\}$.

Proof: The theorem is a special case of the negation-free logic considered in [3]. A formula α is said to be in *negation normal form (NNF)* if $\neg$ appears only in front of atomic formulas. The idea is to reduce validity or logical implication for formulas in NNF to logical implication involving only negation-free formulas. Hence let α be a formula in NNF without $=$ and let $P_1, \ldots P_n$ be the predicate symbols mentioned in α. Let $Q_1, \ldots, Q_n$ be predicate symbols not occurring in α where Q_i and P_i have the same arity. Let α' be α with all occurrences of $\neg P_i$ replaced by Q_i. Note that α' is negation-free. Then it was shown in [3] that α is valid iff $\{\forall(P_1(\vec{x}_1) \vee Q_1(\vec{x}_1)), \ldots, \forall(P_n(\vec{x}_n) \vee Q_n(\vec{x}_n))\} \models \alpha' \vee \bigvee_i (\exists \vec{x}_i . P_i(\vec{x}_i) \wedge Q_i(\vec{x}_i))$. Since the validity problem for formulas in NNF is undecidable, the theorem follows. ∎

Since the query in this theorem is negation-free, it is guaranteed to be in $\mathcal{NF}$. There is also a propositional version of the theorem:

Theorem 8: *The question as to whether KB $\models \alpha$ is co-NP hard, even if KB is of the form $\{(p_1 \vee q_1), \ldots, (p_n \vee q_n)\}$ and α is propositional and negation-free.*

Proof: The proof uses a special case of the reduction of the previous proof. co-NP hardness follows from the fact that logical implication is co-NP hard in the propositional case. ∎

So the problem with including clauses in the KB is not due to *negation* and so will not be resolved by restricting the queries to be in $\mathcal{NF}$. Note, for example, that in both cases the KB is already closed under Resolution. This means that any attempt to limit reasoning by changing how negation will be understood will not work here. As noted above, this is what tautological entailment does: we fail to conclude q from p and $(\neg p \vee q)$ precisely because we admit setups that support both p and its negation. Without negation, tautological entailment and classical entailment coincide:

Theorem 9: *Suppose that KB is a set of formulas $\forall(e \supset c)$ where each c is negation-free, and α is negation-free. Then*

$$\mathcal{E} \cup KB \models \alpha \quad \textit{iff} \quad KB \longrightarrow \alpha$$

Proof: To prove the only-if direction, let $\mathcal{E} \cup$ KB $\models \alpha$ and suppose $s \models^t KB$. Let s^- be s with all negative literals removed. It is easy to prove by induction that s and s^- agree on all negation-free sentences. Hence, in particular, $s^- \models^t KB$. Now let M be a standard model such that for all closed atoms $\rho\theta$, $M \models \rho\theta$ iff $\rho\theta \in s^-$. Again, it is easy to prove by induction that $M \models \gamma$ iff $s^- \models^t \gamma$ for all negation-free sentences γ. Then $M \models KB$ and hence, by assumption, $M \models \alpha$ (standard models clearly satisfy $\mathcal{E}$). Since α is negation-free, we also have $s^- \models^t \alpha$ from which $s \models^t \alpha$ follows.

Conversely, let $KB \longrightarrow \alpha$ and let $M \models \mathcal{E} \cup KB$. It was shown in [8] (proof of Theorem 2) that there is a standard model M' such that M and M' agree on all sentences mentioning only constants in KB and α. Thus, in particular, $M' \models KB$. Now let s be the setup such that $l \in s$ iff $M' \models l$ for all closed literals l. A simple induction proves that for all sentences γ, $M' \models \gamma$ iff $s \models^t \gamma$. Hence $s \models^t KB$ and thus, by assumption, $s \models^t \alpha$. Then $M' \models \alpha$ and, since M' and M agree on sentences mentioning only constants in KB and α, $M \models \alpha$. ∎

So in a sense, tautological entailment does not do enough (it fails to do simple *Modus Ponens* even over Horn clauses), and in another sense, it tries to do far too much (it is subject to the intractability theorems above). It is significant that previous work proposing a limited form of reasoning based on tautological entailment for a KB with disjunction [7, 5] only worked in the propositional case and when the query was in CNF. The first-order case later studied in [10] and [6] required considerable machinery beyond tautological entailment. Moreover, this additional effort only resulted in fewer inferences compared to tautological entailment and hence has only limited appeal.

3.2 Doing more and less than tautological entailment

In the next section, we will propose a new evaluation-based reasoning procedure X that works with disjunctions in the KB, agrees with V when the KB is proper, is reasonably efficient, logically sound, and sometimes even logically complete. It is based on the observation that although disjunction can be used in many ways, it has two major applications: (1) to represent *rules* such as Horn clauses, where we

may need to perform chaining in the reasoning; and (2), to represent *incomplete knowledge* about some individual(s), where we may need to split cases in the reasoning. As argued above, tautological entailment does not do (1) at all, but like classical logic, attempts to handle (2).

As it turns out, we believe that (2) is the computational problem. To see why, consider the following example KBs:

KB1	KB2
$(P(a) \vee P(e) \vee P(f))$	$(P(a) \vee Q(e) \vee Q(c))$
$(P(a) \vee P(e) \vee Q(f))$	$(Q(d) \vee P(b) \vee Q(a))$
$(P(a) \vee Q(e) \vee P(c))$	$(P(a) \vee P(e) \vee P(f))$
$(P(a) \vee Q(e) \vee Q(c))$	$(P(c) \vee Q(e) \vee P(a))$
$(Q(a) \vee P(b) \vee P(d))$	$(Q(a) \vee Q(b) \vee Q(g))$
$(Q(a) \vee P(b) \vee Q(c))$	$(P(a) \vee P(e) \vee Q(f))$
$(Q(a) \vee Q(b) \vee P(g))$	$(Q(b) \vee Q(a) \vee P(g))$
$(Q(a) \vee Q(b) \vee Q(g))$	$(Q(a) \vee P(d) \vee P(b))$

The reader is invited to confirm that one and only one of these logically entails (and hence tautologically entails) $\exists x.(P(x) \wedge Q(x))$. So being required to handle (2) is also being required to solve combinatorial puzzles like this automatically.

Our approach will be to preserve (1), but to give up on (2). To deal with (2) in a limited way, X will split disjunctions and reason by cases, but the number of such splits will be limited by the structure of the query. For example, if we have the clause $(P(a) \vee P(b))$ in the KB, X will infer $(P(a) \vee P(b) \vee P(c))$ and $\exists x.P(x)$ since both of these queries will allow us to split the disjunction. However, if we also have $(\neg P(a) \vee P(b))$ in the KB, then X may be unable to infer $P(b)$, since this query will not allow us to split a disjunction. This restriction will force us into logically incomplete reasoning since, among other things, X will be unable to solve the puzzle above. This is a feature, not a bug.

4 Retrieval with clauses

We begin by considering X_0, a simplified version of X, where the knowledge base is restricted to a (possibly infinite) set of ground clauses. This has the advantage that the basic principles underlying X can be illustrated in a conceptually simpler setting.

Different from V, X_0 returns only two values 0 or 1 where 1 means "known to be true," as before, but 0 now means "not known to be true" instead of "known to be false" in the case of V. As we will see, this change is necessary because, similar to tautological entailment, we want to allow inconsistent knowledge bases. In particular, this means that X_0 should sometimes return 1 for both α and $\neg\alpha$, which is incompatible with the 3-valued answers of V. Note, however, that we are not really losing anything in terms of expressiveness compared to V. While for V a single query α suffices to establish that α is known to be true or known to be false, we now need to pose two queries, α and $\neg\alpha$, to have the same effect.

Definition 7: Let S be a set of ground clauses. Then $\mathrm{UP}(S)$ is the least set which contains S and if $\{l\} \cup c$ and $\bar{l}$ are in $\mathrm{UP}(S)$, then so is c. In other words, $\mathrm{UP}(S)$ is the closure of S under *unit propagation*.

Let S be a non-empty, not necessarily consistent and not necessarily finite, set of ground clauses. Then X_0 is defined as follows:

1. $X_0[S, l] = \begin{cases} 1 & \text{if } l \in \mathrm{UP}(S),\ l \text{ a literal} \\ 0 & \text{otherwise} \end{cases}$
2. $X_0[S, t = t'] = 1$ if t is identical to t', and 0 otherwise
3. $X_0[S, \neg(t = t')] = 1 - X_0[S, t = t']$
4. $X_0[S, \neg\neg\alpha] = X_0[S, \alpha]$
5. $X_0[S, \alpha \vee \beta] =$
$$\begin{cases} 1 & \text{if for some } c \in S \text{ and for all } l \in c, \\ & X_0[S \cup \{l\}, \alpha] = 1 \text{ or } X_0[S \cup \{l\}, \beta] = 1, \\ 0 & \text{otherwise} \end{cases}$$
6. $X_0[S, \neg(\alpha \vee \beta)] = \min\{X_0[S, \neg\alpha], X_0[S, \neg\beta]\}$
7. $X_0[S, \exists x.\alpha] =$
$$\begin{cases} 1 & \text{if for some } c \in S \text{ and for all } l \in c, \\ & \text{there is a } d \in C \text{ such that } X_0[S \cup \{l\}, \alpha^x_d] = 1 \\ 0 & \text{otherwise} \end{cases}$$
8. $X_0[S, \neg\exists x.\alpha] = \min_{d \in C}\{X_0[S, \neg\alpha^x_d]\}$

Besides the fact that the definition of X_0 is somewhat more long-winded than V, the main differences lie in the treatment of literals, disjunctions, and existentials.

When it comes to establishing whether a literal can be inferred, the use of $\mathrm{UP}(S)$ ensures simple applications of *Modus Ponens*. For example, if S contains p, $\neg s$, $(\neg p \vee q)$, and $(\neg q \vee r \vee s)$, then $X_0[S, r] = 1$.

Whenever a disjunction is encountered in the query, Rule 5 allows reasoning by cases, but only with respect to a single clause in S. The same is true for existential quantifiers using Rule 7.

To get a better feel for what can and cannot be inferred using X_0, let us consider the following examples. Suppose

$$S = \{(P(a) \vee Q(b)), (\neg P(a) \vee Q(b))\}$$

Then $X_0[S, \exists x.Q(x)] = 1$ because the existential quantifier in the query allows us to split a clause. If we choose the first clause, we obtain $X_0[S \cup \{P(a)\}, Q(b)] = 1$ because $Q(b) \in \mathrm{UP}(S \cup \{P(a)\})$ and $X_0[S \cup \{Q(b)\}, Q(b)] = 1$, from which the existential follows. On the other hand,

$X_0[S, Q(b)] = 0$ because no splitting of clauses is allowed and $Q(b)$ is not in UP(S), which is the same as S. Now let

$$S = \{(P(a) \vee P(b)), Q(a), Q(b)\}$$

and let $\alpha = \exists x.P(x) \wedge Q(x)$. Then X_0 correctly determines that α follows from S, that is, $X_0[S, \alpha] = 1$. This is because the first clause can be used to split cases when evaluating the existential.

Finally, if we let S be *KB*2 from Section 3, then $X_0[S, \alpha] = 0$ even though *KB*2 logically entails α. The reason is that it is simply not enough to reason by cases with respect to only one clause (when evaluating $\exists$) to solve the puzzle.

As noted already at the beginning of this section, X_0, in contrast to V, applies to knowledge bases which may be logically inconsistent. To see how inconsistencies are dealt with, let us consider two examples. Let

$$S_1 = \{p, q, (q \supset \neg p)\}.$$

S_1 is clearly inconsistent and $X_0[S_1, p \wedge \neg p] = 1$ since both p and $\neg p$ are in UP(S_1). Now consider

$$S_2 = \{(p \vee q), (\neg p \vee q), (p \vee \neg q), (\neg p \vee \neg q)\}.$$

While S_2 is again inconsistent, this time $X_0[S_2, p \wedge \neg p] = 0$ since UP(S_2) $= S_2$. Intuitively, X_0 will only discover shallow inconsistencies, in particular those which it can discover using unit propagation and limited forms of reasoning by cases, depending on the occurrence of disjunctions or existential quantifiers in the query. In a sense, the treatment of inconsistencies by X_0 is more powerful than tautological entailment (for example, S_1 does not tautologically entail $(p \wedge \neg p)$), yet it is still far less powerful than classical logical reasoning, which it needs to be to make it computationally viable.

We conclude our discussion of X_0 by showing that, in the case of proper knowledge bases, X_0 is upward compatible with V in that X_0 returns 1 as an answer just in case V does. The main reason for the agreement is that proper *KB*'s consist only of literals and hence there are no clauses to split.

Theorem 10: *Let KB be non-empty and proper and let* $\mathit{Lits(KB)} = \{l\theta \mid \forall(e \supset l) \in \mathit{KB}$ *and* $\mathcal{E} \models e\theta\}$.
Then for all α, $X_0[\mathit{Lits(KB)}, \alpha] = 1$ *iff* $V[\alpha] = 1$.

Proof: The proof is by induction on the structure of α. Let $\rho\theta$ be an atomic formula in *Lits(KB)*. Then $X_0[\mathit{Lits(KB)}, \rho\theta] = 1$ iff $\rho\theta \in \mathit{Lits(KB)}$ iff $V[\rho\theta] = 1$ by definition and the fact that for any ewff e and substitution θ, $V[e\theta] = 1$ iff $\mathcal{E} \models e\theta$ (Lemma 7 of [8]). Similarly, $X_0[\mathit{Lits(KB)}, \neg\rho\theta] = 1$ iff $\neg\rho\theta \in \mathit{Lits(KB)}$ iff $V[\rho\theta] = 0$ iff $V[\neg\rho\theta] = 1$.

The proof for $(t = t')$ and $\neg(t = t')$ follows immediately because the definitions of X_0 and V agree in this case.

$X_0[\mathit{Lits(KB)}, \neg\neg\alpha] = 1$ iff $X_0[\mathit{Lits(KB)}, \alpha] = 1$ iff (by induction) $V[\alpha] = 1$ iff $V[\neg\neg\alpha] = 1$.

Let $X_0[\mathit{Lits(KB)}, (\alpha \vee \beta)] = 1$. Since *Lits(KB)* consists only of literals, there is an $l \in \mathit{Lits(KB)}$ such that $X_0[\mathit{Lits(KB)} \cup \{l\}, \alpha] = 1$ or $X_0[\mathit{Lits(KB)} \cup \{l\}, \beta] = 1$. By induction and since $\mathit{Lits(KB)} = \mathit{Lits(KB)} \cup \{l\}$, $V[\alpha] = 1$ or $V[\beta] = 1$, from which $V[(\alpha \vee \beta)] = 1$ follows.

Conversely, Let $V[(\alpha \vee \beta)] = 1$. Then $V[\alpha] = 1$ or $V[\beta] = 1$ and hence, by induction, $X_0[\mathit{Lits(KB)}, \alpha] = 1$ or $X_0[\mathit{Lits(KB)}, \beta] = 1$. Let l be any literal in *Lits(KB)*. Then $X_0[\mathit{Lits(KB)} \cup \{l\}, \alpha] = 1$ or $X_0[\mathit{Lits(KB)} \cup \{l\}, \beta] = 1$, from which $X_0[\mathit{Lits(KB)}, (\alpha \vee \beta)] = 1$ follows by definition.

Let $X_0[\mathit{Lits(KB)}, \neg(\alpha \vee \beta)] = 1$ iff $X_0[\mathit{Lits(KB)}, \neg\alpha] = 1$ and $X_0[\mathit{Lits(KB)}, \neg\beta] = 1$ iff (by induction) $V[\alpha] = 0$ and $V[\beta] = 0$ iff $V[\neg\alpha] = 1$ and $V[\neg\beta] = 1$ iff $V[\neg(\alpha \vee \beta)] = 1$.

Let $X_0[\mathit{Lits(KB)}, \exists x.\alpha] = 1$. Then for some literal $l \in \mathit{Lits(KB)}$ and $c \in \mathcal{C}$, $X_0[\mathit{Lits(KB)} \cup \{l\}, \alpha^x_d] = 1$. By induction and since $\mathit{Lits(KB)} = \mathit{Lits(KB)} \cup \{l\}$, $V[\alpha^x_d] = 1$. By Theorem 9 of [8], there is a $d' \in H_1^+$ such that $V[\alpha^x_{d'}] = 1$. Therefore, $V[\exists x.\alpha] = 1$.

Conversely, let $V[\exists x.\alpha] = 1$. Then for some $d \in H_1^+$, $V[\alpha^x_d] = 1$ and, by induction, $X_0[\mathit{Lits(KB)}, \alpha^x_d] = 1$. Then for any literal l, $X_0[\mathit{Lits(KB)} \cup \{l\}, \alpha^x_d] = 1$, from which $X_0[\mathit{Lits(KB)}, \exists x.\alpha] = 1$ follows.

The case $\neg\exists x.\alpha$ follows easily by induction. ∎

While X_0 is clearly limited in what it does compared to logical entailment, more needs to be done to ensure its practicality. For one, we want to allow more than just ground clauses in the *KB*. For another, Rules 5, 7, and 8 appeal to substitutions over an infinite domain, which needs to be constrained.

To go beyond ground clauses, we first generalize the notion of a proper *KB* introduced in Section 2. The idea is that instead of universally quantified literals, possibly restricted by ewffs, we now allow arbitrary clauses to take the place of the literals.

Definition 8 Let e be an ewff and c a disjunction of literals whose arguments are distinct variables. Then $\forall(e \supset c)$ is called a *∀-clause*.

Definition 9: A *KB* is called *proper*$^+$ if *KB* is a finite non-empty collection of ∀-clauses. Given a proper$^+$ *KB*, *gnd(KB)* is defined as $\{c\theta \mid \forall(e \supset c) \in \mathit{KB}$ and $\mathcal{E} \models e\theta\}$.

Note that, under the standard interpretations introduced in Section 2, a proper$^+$ KB is a finite representation of the (usually infinite) set of ground clauses gnd(KB). In particular, a standard interpretation satisfies KB iff it satisfies gnd(KB). We are now ready to define X for proper$^+$ KB's and arbitrary α.

1. $X[KB, l] = \begin{cases} 1 & \text{if } l \in \text{UP}(\text{gnd}(KB)), \ l \text{ a literal} \\ 0 & \text{otherwise} \end{cases}$
2. $X[KB, t = t'] = 1$ if t is identical to t', and 0 o.w.
3. $X[KB, \neg(t = t')] = 1 - X[KB, t = t']$
4. $X[KB, \neg\neg\alpha] = X[KB, \alpha]$
5. $X[KB, \alpha \vee \beta] =$
$\begin{cases} 1 & \text{there is a } \forall(e \supset c) \in KB \text{ and a } \theta \in H_k^+ \text{ such that}^4 \\ & X[KB, e\theta] = 1 \text{ and for all } l \in c, \\ & X[KB \cup \{l\theta\}, \alpha] = 1 \text{ or } X[KB \cup \{l\theta\}, \beta] = 1, \\ & \text{where } k \text{ is the number of free variables in } c \\ 0 & \text{otherwise} \end{cases}$
6. $X[KB, \neg(\alpha \vee \beta)] = \min\{X[KB, \neg\alpha], X[KB, \neg\beta]\}$
7. $X[KB, \exists x.\alpha] =$
$\begin{cases} 1 & \text{if there is a } \forall(e \supset c) \in KB \text{ and a } \theta \in H_k^+ \text{ such that} \\ & X[KB, e\theta] = 1 \text{ and for all } l \in c \\ & \text{there is a } d \in H_{k+1}^+ \text{ such that } X[KB \cup \{l\theta\}, \alpha_d^x] = 1, \\ & \text{where } k \text{ is as above.} \\ 0 & \text{otherwise} \end{cases}$
8. $X[KB, \neg\exists x.\alpha] = \min_{d \in H_1^+}\{X[KB, \neg\alpha_d^x]\}$

Using X with a KB is very similar to using X_0 with gnd(KB). The main difference is the way disjunctions and existentials are handled. While in X_0 we allow any clause from S to be chosen for case splitting, in X we can only split substitution instances over H_k^+ of a $\forall$-clause of KB. Also, X_0 evaluates negated existentials (universals) with respect to a finite set of constants, just like V does. Nevertheless, this is done without loss of generality in that X and X_0 agree on their answers. To prove this we need the following lemmas.

Let $*$ be any bijection from $\mathcal{C}$ to $\mathcal{C}$. Let α^* mean α with c replaced by c^*. Let S^* mean $\{\alpha^* \mid \alpha \in S\}$. Let θ^* be the substitution which assigns variable x the value c^* if θ assigns x the value c.

Lemma 11: $X_0[S, \alpha] = X_0[S^*, \alpha^*]$.

Lemma 12: $c \in UP(S)$ *iff* $c^* \in UP(S^*)$.

The lemmas are proved by simple induction arguments over the structure of α and the length of a derivation by unit propagation, respectively.

[4] Here, $\theta \in H_k^+$ means that the substitutions range only over the elements in H_k^+.

Lemma 13: $gnd(KB)^* = gnd(KB^*)$.

Proof: Let $c' \in \text{gnd}(KB)^*$. Then $c' = c^*\theta^*$ for some c and θ such that $c\theta \in \text{gnd}(KB)$ with $\forall(e \supset c) \in KB$ and $\mathcal{E} \models e\theta$. A simple induction shows that for any ewff e, $\mathcal{E} \models e$ iff $\mathcal{E} \models e^*$. Hence $\mathcal{E} \models e^*\theta^*$. Since $\forall(e^* \supset c^*) \in KB^*$, we have that $c^*\theta^* \in \text{gnd}(KB^*)$, that is, $c' \in \text{gnd}(KB^*)$. The converse is proved analogously. ∎

Theorem 14: *For any proper$^+$ KB and any sentence α, $X[KB, \alpha] = 1$ iff $X_0[gnd(KB), \alpha] = 1$.*

Proof: Note that X and X_0 already agree on all rules except 5, 7, and 8. We show that these rules can also be made identical in the case of proper$^+$ KBs. The theorem then follows by a simple induction argument.

It suffices to show the following:

1. $X_0[\text{gnd}(KB), \neg\exists x.\alpha] = \min_{d \in H_1^+}\{X_0[\text{gnd}(KB), \neg\alpha_d^x]\}$

2. If $X_0[\text{gnd}(KB), (\alpha \vee \beta)] = 1$ then for some $\forall(e \supset c) \in KB$ and $\theta \in H_k^+$ s.t. $X_0[\text{gnd}(KB), e\theta] = 1$ and for all $l \in c$, $X_0[\text{gnd}(KB) \cup \{l\theta\}, \alpha] = 1$ or $X_0[\text{gnd}(KB) \cup \{l\theta\}, \beta] = 1$.

3. If $X_0[\text{gnd}(KB), \exists x.\alpha] = 1$ then for some $\forall(e \supset c) \in KB$ and $\theta \in H_k^+$ s.t. $X_0[\text{gnd}(KB), e\theta] = 1$ and for all $l \in c$ there is a $d \in H_{k+1}^+$ such that $X_0[\text{gnd}(KB) \cup \{l\theta\}, \alpha_d^x] = 1$.

We now prove these cases in turn.

1. Suppose $X_0[\text{gnd}(KB), \neg\alpha_d^x] = 1$ for all $d \in H_1^+$. We need to show that $X_0[\text{gnd}(KB), \neg\alpha_d^x] = 1$ for all $d \in \mathcal{C}$. Let $d \notin H_1^+$ and let d' be the constant in H_1^+ which does not appear in $KB \cup \{\alpha\}$. Let $*$ be a bijection which swaps d and d' and is the identity otherwise. Since $X_0[\text{gnd}(KB), \neg\alpha_{d'}^x] = 1$ by assumption, $X_0[\text{gnd}(KB)^*, \neg\alpha_{d'}^{x*}] = 1$ by Lemma 11. Since, by Lemma 13, $\text{gnd}(KB)^* = \text{gnd}(KB^*)$ and $KB^* = KB$, $\alpha^* = \alpha$, and $d = d'^*$ by the definition of $*$, $X_0[\text{gnd}(KB), \neg\alpha_d^x] = 1$ follows.

2. Let $X_0[\text{gnd}(KB), (\alpha \vee \beta)] = 1$. By definition, there is a $c\theta \in \text{gnd}(KB)$ s.t. $\forall(e \supset c) \in KB$ and $\mathcal{E} \models e\theta$ and for all $l \in c$, $X_0[\text{gnd}(KB) \cup \{l\theta\}, \alpha] = 1$ or $X_0[\text{gnd}(KB) \cup \{l\theta\}, \beta] = 1$. Suppose c contains k variables. Then $c\theta$ mentions at most k names not in H_k^+. Let these be $d_1, \ldots, d_m$ with $m \leq k$. Let $*$ be the bijection which is the identity everywhere except that it swaps d_i with d_i' where the d_i' are distinct elements of H_k^+ which do not appear in $KB \cup \{(\alpha \vee \beta)\}$. (These must exist given the definition of H_k^+.) Given the choice of $*$ and since the values which θ assigns to variables not occurring in c can be chosen arbitrarily, we can assume, without loss of generality, that $\theta^* \in H_k^+$. Also, by Lemma 11, $\mathcal{E} \models e^*\theta^*$ and for all $l^* \in c^*$, $X_0[(\text{gnd}(KB) \cup \{l\theta\})^*, \alpha^*] = 1$ or $X_0[(\text{gnd}(KB) \cup \{l\theta\})^*, \beta^*] = 1$. Since $\alpha = \alpha^*$, $\beta = \beta^*$,

$c = c^*$, $e = e^*$, $\text{gnd}(KB) = \text{gnd}(KB)^*$, we have that $X_0[\text{gnd}(KB), e\theta^*] = 1$ and for all $l \in c$, $X_0[\text{gnd}(KB) \cup \{l\theta^*\}, \alpha] = 1$ or $X_0[\text{gnd}(KB) \cup \{l\theta^*\}, \beta] = 1$.

3. Let $X_0[\text{gnd}(KB), \exists x.\alpha] = 1$. Using an argument completely analogous to the previous case one can show that there is an $\forall(e \supset c) \in KB$ and a $\theta \in H_k^+$ such that $X_0[\text{gnd}(KB), e\theta] = 1$ and for all $l \in c$ there is a $c \in \mathcal{C}$ such that $X_0[\text{gnd}(KB) \cup \{l\theta\}, \alpha_d^x] = 1$. All that is left to show is that d can be chosen from H_{k+1}^+. Suppose d is not in H_{k+1}^+, then consider a bijection $*$ which is the identity everywhere except that it swaps d with d' where $d' \in H_{k+1}^+$ and does not appear in either KB, $l\theta$, or α. Similar to the previous case it then follows that $X_0[\text{gnd}(KB) \cup \{l\theta\}, \alpha_{d'}^x] = 1$ and we are done. ∎

5 Advantages of X

But what does X have going for it, besides its close relationship with X_0? In particular, what can we say about X as a method of logical inference? Here we will show that X always performs logically sound reasoning, that it is fully compatible with V, and that reasoning as defined by X is decidable.

Before showing that X always performs sound reasoning, we need to adapt our original definition of soundness and completeness (Definition 4) to the case of two-valued evaluation methods:

Definition 10: Let $S, T \subseteq \mathcal{L}$, and let $f \in [\mathcal{L} \to \{0,1\}]$. Then

- f is logically sound wrt S for T iff for every $\alpha \in T$, if $f[\alpha] = 1$ then $S \models \alpha$.
- f is logically complete wrt S for T iff for every $\alpha \in T$, if $S \models \alpha$ then $f[\alpha] = 1$.

We need the following simple properties to prove soundness. Since X has already been defined for closed atomic ewffs, we obtain by a simple induction:

Lemma 15: *For any proper$^+$ KB, ewff e and substitution θ, $X[KB, e\theta] = 1$ iff $\mathcal{E} \models e\theta$.*

We also use the following fact about logical entailment:

Lemma 16: *Let S be any set of sentences, and $c = (l_1, \ldots, l_n)$ a ground clause. Then for any sentence α, $S \cup \{c\} \models \alpha$ iff $S \cup \{l_i\} \models \alpha$ for all l_i.*

Theorem 17: *X is logically sound wrt to any proper$^+$ KB for $\mathcal{L}$.*

Proof: The proof is by induction on the structure of α. Let $X[KB, l] = 1$ for a literal l. Then $l \in \text{UP}(\text{gnd}(KB))$. Thus $\mathcal{E} \cup \text{UP}(\text{gnd}(KB)) \models l$ and, since $\mathcal{E} \cup \text{UP}(\text{gnd}(KB))$ and $\mathcal{E} \cup KB$ are logically equivalent, $\mathcal{E} \cup KB \models l$. If $X[KB, t = t'] = 1$ then t and t' are identical, which is clearly sound, and similarly for $X[KB, \neg(t = t')] = 1$.

The cases $\neg\neg\alpha$ and $\neg(\alpha \vee \beta)$ follow easily by induction.

Let $X[KB, (\alpha \vee \beta)] = 1$. Then for some $\forall(e \supset c) \in KB$ and $\theta \in H_k^+$ with $X[KB, e\theta] = 1$ we have that for all $l \in c$, $X[KB \cup \{l\theta\}, \alpha] = 1$ or , $X[KB \cup \{l\theta\}, \beta] = 1$. Hence, by induction, $\mathcal{E} \cup KB \cup \{l\theta\} \models \alpha$ or $\mathcal{E} \cup KB \cup \{l\theta\} \models \beta$ for all $l \in c$. Therefore, for all $l \in c$, $\mathcal{E} \cup KB \cup \{l\theta\} \models (\alpha \vee \beta)$. By Lemma 16 we have that $\mathcal{E} \cup KB \cup \{c\theta\} \models (\alpha \vee \beta)$. Since $X[KB, e\theta] = 1$, we also have $\mathcal{E} \models e\theta$ by Lemma 15, that is, $c\theta \in \text{UP}(\text{gnd}(KB))$. Therefore, since $\mathcal{E} \cup KB \cup \{c\theta\}$ and $\mathcal{E} \cup KB$ are logically equivalent, $\mathcal{E} \cup KB \models (\alpha \vee \beta)$ follows.

The proof for $\exists x.\alpha$ is completely analogous to the previous case.

Finally, let $X[KB, \neg\exists x.\alpha] = 1$. Then $X[KB, \neg\alpha_d^x] = 1$ for all $d \in H_1^+$. Then, by induction, $\mathcal{E} \cup KB \models \neg\alpha_d^x$ for all $d \in H_1^+$. By Theorem 2, $\mathcal{E} \cup KB \models \neg\alpha_d^x$ for all $d \in \mathcal{C}$ and thus $\mathcal{E} \cup KB \models \neg\exists x.\alpha$. ∎

So if X ever returns 1, we know that the query is logically entailed by $(\mathcal{E} \cup KB)$.

Furthermore, if the KB is proper, that is, if all the clauses are unit clauses, then X agrees with V, which is an immediate consequence of Theorem 10 and 14.

Corollary 18: *If KB is proper then for any sentence α, $X[KB, \alpha] = 1$ iff $V[\alpha] = 1$.*

Intuitively, this is so because for a proper KB there are no clauses to split, and the rules of X for $\vee$ and $\exists$ essentially reduce to those of V. So in this case, X will perform logically sound and complete reasoning for queries in $\mathcal{NF}$, just like V, and just as efficiently.

Unlike V however, X does allow disjunctions in the KB. It will not necessarily perform logically complete reasoning over them, but many simple cases can be handled, as discussed in the examples above for X_0. As with X_0, the X procedure is able to perform chaining, but now using quantifiers and equality. For example, if a and b are distinct constants, and

$$KB = \{\, \forall(x \neq a \supset P(x)),\ \forall(x = y \supset \neg P(x) \vee Q(y)) \,\},$$

then $X[KB, Q(b)] = 1$. The X procedure also performs case splitting, including some that would have required an infinite number of splits in X_0. For example, suppose that

$$KB = \{\, \forall(P(x, a) \vee P(x, b)) \,\}.$$

Then we have that $X[KB, \forall x \exists y.P(x,y)] = 1$. This example also shows that it would not have worked to define X to first choose n clauses to split (where perhaps n is the number of connectives in the query), and then to use V on the resulting augmented *KB*.

The final desirable feature of X is its computability:

Theorem 19: *X is decidable.*

The argument is as follows: Since the *KB* and H_k^+ are both finite, there are only finitely many choices to consider in splitting cases. Moreover, and similar to V, only finitely many constants need to be considered as substitutions for an existentially quantified variable. So all that is left to show decidability is argue that for any ground literal l, it is always possible to decide if it is a member of UP(gnd(*KB*)). Because we are using unit propagation and not full Resolution, this is similar (ignoring equality) to deciding if a ground atom is entailed by a collection of quantified Horn clauses. This can be done since there are no function symbols.

6 Conclusions

While the evaluation method V considered in previous work has many advantages in that it provides an efficient, logically sound, and sometimes complete reasoning method, its perhaps greatest shortcoming is that it only applies to proper knowledge bases, that is, sets of literals. In this paper we proposed X, which is compatible with V and, in addition, allows *KB*'s to contain arbitrary clauses. The main challenge in developing X has been to constrain it in such a way that reasoning remains decidable and, at the same time, allows for interesting inferences such as limited applications of *Modus ponens*. We have seen that on both counts X is preferable over limited reasoning methods like tautological entailment.

Many future questions remain to be explored. For example, while we have results concerning the computability of X, a stronger argument needs to be made concerning its practicality. In normal cases, we would expect X to be *efficiently* computable. It is true that the procedure can split a clause for each disjunction or existential in the query, but this would be worst-case exponential only in the number of disjunctions and existentials in the query (and with some effort, we believe this can be made exponential in the depth of connectives only). As in [8] we would normally expect the query to be at most logarithmic in the size of the *KB*, and the depth of connectives to be be logarithmic in the size of the query. Finally, as suggested above, testing membership of a ground literal in UP(gnd(*KB*)) is comparable to query answering over Horn clauses with a finite Herbrand universe, that is, Datalog [1]. Nevertheless, it still needs to be shown that we can indeed deal effectively with a KB containing (say) a large number of atomic facts and a few disjunctions. Also, can we implement X in such a way that in cases where disjunction is not needed, standard database techniques (like projections, joins *etc.*) can be used?

On the logical side, we have not attempted to characterize useful cases of queries and KBs for which X is logically complete, beyond those of V: proper KBs with queries that are in normal form. We do believe that X computes precisely the entailment relation for a new *limited logic* that is interestingly different from tautological entailment. We will characterize this logic in a separate publication.

References

[1] Abiteboul, S., Hull, R., and Vianu, V., *Foundations of Databases*. Addison-Wesley, Reading, MA, 1995.

[2] Anderson, A. R. and Belnap, N. D., *Entailment, The Logic of Relevance and Necessity*. Princeton University Press, 1975.

[3] Delgrande, J. and Gupta, A., The complexity of minimum truth assignments and implication in negation-free formulae. *Annals of Mathematics and Artificial Intelligence* **18**, 1996, pp. 51–67.

[4] Dunn, J. M., Intuitive Semantics for First-Degree Entailments and Coupled Trees. *Philosophical Studies* **29**, 1976, pp. 149–168.

[5] Frisch, A., Inference without chaining. *Proceedings of IJCAI-87*, Morgan Kaufmann, San Francisco, pp. 515–519, 1987.

[6] Lakemeyer, G., *Models of belief for decidable reasoning in incomplete knowledge bases*. Ph. D. thesis, Dept. of Computer Science, University of Toronto, Toronto, Ont., 1990.

[7] Levesque, H., A logic of implicit and explicit belief. in *Proc. AAAI-84*, pp. 198–202, Austin, TX, 1984.

[8] Levesque, H. J., A completeness result for reasoning with incomplete first-order knowledge bases. *Proceeding of the 6th International Conference on Knowledge Representation and Reasoning (KR'98)*, Morgan Kaufmann, San Francisco, 1998, pp. 14–23.

[9] Levesque, H. J. and Lakemeyer, G., *The Logic of Knowledge Bases*. MIT Press, 2001.

[10] Patel-Schneider, P., A decidable first-order logic for knowledge representation. *Proceedings of IJCAI-85*, Los Angeles, CA, pp. 455–458, 1985.

The Complexity of Model Checking for Knowledge Update

Chitta Baral*
Dept. of Computer Science & Engineering
Arizona State University
Tempe, AZ 85287, USA

Yan Zhang†
School of Computing & Information Technology
University of Western Sydney
Penrith South DC, NSW 1797, Australia

Abstract

The authors have recently proposed a formal theory of knowledge update based on the semantics of modal logic S5 (Baral & Zhang 2001). In that system, an agent's knowledge set is represented as a S5 formula and update on agent's knowledge is implemented by updating the corresponding Kripke models of the agent's knowledge set. In this paper, we further investigate the computational complexity of model checking for knowledge update. We first show that in general the model checking for knowledge update is Σ_3^P-complete, which places the problem at one layer higher in the polynomial hierarchy than the traditional model based belief update (e.g. PMA). We then identify a subclass of knowledge update problems that has polynomial time complexity for model checking. We point out that some important knowledge update problems belong to this subclass. We also address other subclasses of knowledge update problems for which the complexity of model checking ranges from NP-complete to Σ_2^P-complete.

1 Introduction

Different from traditional belief revision and update, knowledge update is concerned with the issue of changing an agent's knowledge where a modal logic is usually used to specify the agent's knowledge. The authors have recently proposed a formal theory of knowledge update based on the semantics of the modal logic S5 (Baral & Zhang 2001). In that formulation, an agent's knowledge is represented as a S5 formula and updating the agent's knowledge is implemented by updating the corresponding Kripke models of the agent's knowledge (Baral & Zhang 2001). In this paper, we investigate the computational complexity of model checking for knowledge update. We first show that in general the model checking problem for knowledge update is Σ_3^P-complete, which places the problem at one layer higher in the polynomial hierarchy than the traditional model based belief update (e.g. PMA) (Liberatore & Schaerf 2001). We then identify a subclass of knowledge update problems for which model checking can be achieved in polynomial time. We show that some important knowledge update problems belong to this subclass. We also identify other subclasses of knowledge updates for which the complexity for model checking ranges from NP-complete to Σ_2^P-complete.

The rest of this paper is organized as follows. In Section 2 we introduce the necessary preliminaries that are needed in the paper. We first review the semantics of knowledge update, and then summarize the basic concepts and notations of complexity theory. In Section 3 we investigate the complexity of model checking for general knowledge updates, and in Section 4 we identify an important subclass of knowledge update problems for which model checking can be done in polynomial time. In Section 5 we examine other subclasses of knowledge update problems for which the complexity of model checking is lower than the general case, but still intractable (i.e. NP-complete or beyond). We show that several interesting knowledge update cases belong to these subclasses. Finally, in Section 6 we conclude the paper with some remarks.

2 Preliminaries

In this section, we briefly present the formal semantics of knowledge update that was developed by the authors in (Baral & Zhang 2001) and the background

*E-mail: chitta@asu.edu
†Corresponding author. E-mail: yan@cit.uws.edu.au

and terminology needed to understand our complexity results described in this paper. Throughout the paper we restrict our attention to a finite propositional language.

2.1 Knowledge Update: Semantics

Our formalization of knowledge update will be based on the semantics of the propositional modal logic S5 with a single agent. In general, under Kripke semantics, a *Kripke structure* is a triple (W, R, π), where W is a set of possible worlds, R is an equivalence relation on W, and π is a truth assignment function that assigns a propositional valuation to each world in W. Given a Kripke structure $S = (W, R, \pi)$, a *Kripke interpretation* is a pair $M = (S, w)$, where $w \in W$ is referred to as the *actual world* of M. The entailment relation $\models$ between Kripke interpretations and formulas is defined to provide semantics for S5 formulas (Fagin *et al.* 1995).

In the case of single agent, however, we may restrict ourselves to those S5 structures in which the relation R is universal, i.e. each world is accessible from every world, and worlds are identified with the set of atoms true at the worlds (Meyer & van der Hoek 1995). To simplify a comparison between two worlds (e.g. Definition 2), we may view an atom $p \in w$ iff $w \models p$. Therefore, in our context a Kripke structure (W, R, π) is uniquely characterized by W and we simplify a Kripke interpretation as a pair (W, w) which we call a *k-model*, where w indicates the actual world of the agent and W presents all possible worlds that the agent may access. Note that w is in W for any k-model (W, w).

In our following description, we use $a, b, c, p, \cdots$, $x, y, z, \cdots$ to denote primitive propositional atoms; $P, Q, R, X, X, Z, \cdots$ to denote sets of primitive propositional atoms; $\phi, \psi, \upsilon, \cdots$ to denote propositional formulas without including modalities (we also call them *objective* formulas); and $\alpha, \beta, \gamma, \mu, \cdots$ to denote formulas that may contain the modal operator K. For convenience, we also use $T = \alpha_1 \wedge \cdots \wedge \alpha_k$ to represent a finite set of formulas $\{\alpha_1, \cdots, \alpha_k\}$ and call T a *(knowledge) set.*

Definition 1 *(Simplified S5 Semantics) Let P be the set of all primitive propositions in the language. The entailment relation $\models$ under simplified S5 semantics is defined as follows:*

1. $(W, w) \models p$ *iff* p *is primitive (i.e.* $p \in P$*) and* $w \models p$*;*
2. $(W, w) \models \alpha \wedge \beta$ *iff* $(W, w) \models \alpha$ *and* $(W, w) \models \beta$*;*
3. $(W, w) \models \neg\alpha$ *iff it is not the case that* $(W, w) \models \alpha$*;*
4. $(W, w) \models K\alpha$ *iff* $(W, w') \models \alpha$ *for all* $w' \in W$*.*

Given a formula α, $M = (W, w)$ is called a k-model of α if $M \models \alpha$. We use $Mod(\alpha)$ to denote the set of all k-models of α. For an objective formula ϕ, $Mod(\phi)$ simply denotes the set of models of ϕ, and $w \models \phi$ denotes that w is a *model* of ϕ, i.e. w is a world in which ϕ is true.

Now the basic problem of knowledge update that we would like to investigate is described as follows: given a *k-model* $M = (W, w)$, that is viewed as a *knowledge state* of an agent (or simply called a state of the agent), and a formula μ - the agent's new knowledge that may contain modal operator K, how do we update M to another k-model $M' = (W', w')$ such that $M' \models \mu$ and M' is *minimally different* from M with respect to some criterion. To approach this problem, we first need to provide a definition of *closeness* between two k-models with respect to a given k-model.

Definition 2 *(Baral & Zhang 2001) Let $M = (W, w)$, $M_1 = (W_1, w_1)$ and $M_2 = (W_2, w_2)$ be three k-models. We say M_1 is* as close to M as *M_2, denoted as $M_1 \leq_M M_2$, if:*

1. $(w_1 \setminus w \cup w \setminus w_1) \subseteq (w_2 \setminus w \cup w \setminus w_2)$*; or*
2. $w_1 = w_2$ *and one of the following conditions holds:*
   - *(i) if $W \subseteq W_1$, then (a) there exist some ϕ and ψ such that $M \models K\phi$ and $M_2 \not\models K\phi$ and $M \not\models K\psi$ and $M_2 \models K\psi$, or (b) for any ϕ if $M \models K\phi$ and $M_1 \not\models K\phi$, then $M_2 \not\models K\phi$;*
   - *(ii) if $W_1 \subset W$, then condition (a) above is satisfied, or (c) for any ϕ if $M \not\models K\phi$ and $M_1 \models K\phi$, then $M_2 \models K\phi$;*
   - *(iii) if $W \not\subset W_1$ and $W_1 \not\subset W$, then both conditions (b) and (c) above are satisfied.*

We denote $M_1 <_M M_2$ if $M_1 \leq_M M_2$ and $M_2 \not\leq_M M_1$.

In the above definition, condition 1 simply says that the symmetric differences between w and w_1 is not bigger than that between w and w_2, while in condition 2, (i), (ii) and (iii) express that different preferences are applied to compare knowledge between M_1 and M_2 with respect to M. For convenience, given a k-model M, if we denote $KM = \{\phi \mid \text{for all } w \in W, w \models \phi\}$, then (a) is equivalent to $KM \setminus KM_2 \neq \emptyset$ and $KM_2 \setminus KM \neq \emptyset$; (b) is equivalent to $KM \setminus KM_1 \subseteq KM \setminus$

KM_2; and (c) is equivalent to $KM_1 \setminus KM \subseteq KM_2 \setminus KM$. Also (b) and (c) together present a difference on both knowledge decrease and increase between M_1 and M_2 in terms of M. It is also easy to see that $\leq_M$ is a partial ordering.

Note that during the comparison between two k-models, we give preference to the change of the actual world over the change of the knowledge about the world. (The closeness criterion between actual worlds that we use is the commonly used criterion (Winslett 1988) based on symmetric difference). For instance, if the actual world of a k-model M_1 is closer to the actual world of M than the actual world of another k-model M_2, we will think M_1 is closer to M and the comparison of knowledge between M_1 and M_2 is ignored. Only when both M_1 and M_2 have the same actual world, we will compare the knowledge of M_1 and M_2 in terms of M. This seems to be intuitive to us. In fact, the comparison between actual worlds determines the *actual distance* between two k-models to the given k-model M. If M_1 and M_2 have the same actual distance to M, the *knowledge distance* is then taken into account.

Basically, condition (i) (or (ii) resp.) in Definition 2 defines a knowledge preference based on knowledge decrease (or increase resp). That is, if M_1 *only* looses knowledge from M (or *only* gains some knowledge to M, resp.), then M_1 is preferred over those k-models that have both knowledge decrease and increase from M, i.e. (a), and also preferred over those k-models that only loose more knowledge from M (or add more knowledge to M, resp.) than M_1 does, i.e. (b) or (c) respectively. Condition (iii), on the other hand, deals with the mixed situation that M_1 has both knowledge decrease and increase from M. In this case, a combined difference on knowledge decrease and increase is applied to determine the knowledge distance, i.e. (b) *and* (c).

Definition 3 *(Baral & Zhang 2001) Let $M = (W, w)$ be a k-model and μ a formula. A k-model $M' = (W', w')$ is called a* possible resulting k-model *after updating M with μ if and only if the following conditions hold:*

1. *$M' \models \mu$;*
2. *there does not exist another k-model $M'' = (W'', w'')$ such that $M'' \models \mu$ and $M'' <_M M'$.*

We denote the set of all possible resulting k-models after updating M with μ as $Res(M, \mu)$.

Example 1 Let $T = Kc \wedge \neg Ka \wedge \neg Kb \wedge K(a \vee b)$ and $\mu = K\neg c$. We denote

$$w_0 = \{a, b, c\}, w_1 = \{a, c\}, w_2 = \{b, c\},$$
$$w_3 = \{c\}, w_4 = \{a, b\}, w_5 = \{a\},$$
$$w_6 = \{b\}, w_7 = \emptyset.$$

Clearly, $M_0 = (\{w_0, w_1, w_2\}, w_0)$ is a k-model of T. Consider the update of M_0 with μ. Let $M_1 = (\{w_4, w_5, w_6\}, w_4)$. Now we show that M_1 is a possible resulting k-model after updating M_0 with μ.

Since $(w_0 \setminus w_4 \cup w_4 \setminus w_0) = \{c\}$, we first consider any possible k-model $M' = (W', w')$ such that $(w_0 \setminus w' \cup w' \setminus w_0) \subset \{c\}$. Clearly, the only possible w' would be w_0 itself. Let $M' = (W', w_0)$, where W' is a subset of $\{w_0, \cdots, w_7\}$. However, since $c \in w_0$, there does not exist any W' such that $M' \models K\neg c$. Therefore, from Definition 2, only condition 2 can be used to find a possible M' such that $M' <_M M_1$. So we assume $M' = (W', w_4)$. On the other hand, from M_0 and M_1, it is easy to see that $KM_0 = \{c, a \vee b\}$ and $KM_1 = \{\neg c, a \vee b\}$[1]. Then we have $KM_0 \setminus KM_1 = \{c\}$ and $KM_1 \setminus KM_0 = \{\neg c\}$. Ignoring the detailed verifications, we can show that there does not exist such $M' = (W', w_4)$ satisfying $KM_0 \setminus KM' = KM' \setminus KM_0 = \emptyset$. ■

Based on the k-model update, updating a formula (knowledge set) T in terms of another formula μ is then achieved by updating every k-model of T with μ.

Definition 4 *(Baral & Zhang 2001) Let T and μ be two S5 formulas. The* update of T with μ, *denoted as $T \diamond \mu$, is defined by $Mod(T \diamond \mu) = \bigcup_{M \in Mod(T)} Res(M, \mu)$.*

Given a set of k-models $\mathcal{S}$ and a k-model M, let $\leq_M$ be an ordering on $\mathcal{S}$ as we defined in Definition 2. By $Min(\mathcal{S}, \leq_M)$ we mean the set of all elements in $\mathcal{S}$ that are minimal with respect to the ordering $\leq_M$. The following theorem shows that our knowledge update can be characterized by a minimal change criterion based on an ordering on k-models.

Theorem 1 *(Baral & Zhang 2001) Let T and μ be two formulas. Then*

$$Mod(T \diamond \mu) = \bigcup_{M \in Mod(T)} Min(Mod(\mu), \leq_M).$$

2.2 Computational Complexity

We now introduce some basic notions from complexity theory and refer to (Garey & Johnson 1979) for

[1] For simplicity, here we only consider the *prime* formulas ϕ in KM in the sense that if $\phi \in KM$, then there is not another ψ such that $\models \psi \supset \phi$ and $\psi \in KM$.

further details. Two important complexity classes are P and NP. The class of P includes those decision problems solvable by a polynomial-time deterministic Turing machine. The class of NP, on the other hand, consists of those decision problems solvable by a polynomial-time nondeterministic Turing machine.

Let $\mathcal{C}$ be a class of decision problems. The class $P^{\mathcal{C}}$ consists of the problems solvable by a polynomial-time deterministic Truing machine with an oracle for a problem from $\mathcal{C}$, while the class $NP^{\mathcal{C}}$ includes the problems solvable by a nondeterministic Turing machine with an oracle for a problem in $\mathcal{C}$. By co-$\mathcal{C}$ we mean the class consisting of the complements of the problems in $\mathcal{C}$.

The classes Σ_k^P and Π_k^P of the *polynomial hierarchy* are defined as follows:

$$\Sigma_0^P = \Pi_0^P = P, \text{ and}$$
$$\Sigma_k^P = NP^{\Sigma_{k-1}^P},\ \Pi_k^P = \text{co-}\Sigma_k^P \text{ for all } k > 1.$$

It is easy to see that $NP = \Sigma_1^P$ and co-$NP = \Pi_1^P$. A problem A is *complete* for a class $\mathcal{C}$ if $A \in \mathcal{C}$ and for every problem B in $\mathcal{C}$ there is a polynomial transformation of B to A.

The prototypical Σ_k^P-complete and Π_k^P-complete problems are deciding the validity of quantified Boolean formulas (QBFs) of the form:

$$Q_1 X_1 Q_2 X_2 \cdots Q_k X_k E, k \geq 1, \qquad (1)$$

where E is a Boolean expression using propositional atoms over alphabets $X_1, X_2, \cdots$, and X_k, and the Q_i's are alternating qualifiers from $\{\forall, \exists\}$ $(1 \leq i \leq k)$. If $Q_1 = \exists$, then deciding the validity of (1) is Σ_k^P-complete, while deciding the validity of (1) is Π_k^P-complete if $Q_1 = \forall$.

Let X and Y be two finite set of propositional atoms where X and Y have the same cardinality, i.e. $|X| = |Y|$. For convenience, we use notion $X \equiv Y$ to stand for formula $(x_1 \equiv y_1) \wedge (x_2 \equiv y_2) \wedge \cdots \wedge (x_m \equiv y_m)$. Consequently, $X \equiv \neg Y$ stands for formula $(x_1 \equiv \neg y_1) \wedge (x_2 \equiv \neg y_2) \wedge \cdots \wedge (x_m \equiv \neg y_m)$. We also use $\neg X$ to denote the set $\{\neg x_i \mid x_i \in X\}$ (or formula $\bigwedge_{x_i \in X} \neg x_i$), and use notion $\bigvee \neg X$ to stand for formula $\bigvee_{x_i \in X} \neg x_i$. For a given formula α, we use $|\alpha|$ to denote the length of α.

The problem of *model checking* for knowledge update is described as follows: Given a knowledge set T, a formula μ, and a k-model M, deciding whether $M \in Mod(T \diamond \mu)$. It is well known that the model checking problem for traditional belief revision and update is located at the lower end of the polynomial hierarchy from P to Σ_2^P depending on specific revision/update operators and additional restrictions (if any) (Liberatore & Schaerf 2001).

3 The General Case

In this section, we investigate the complexity of model checking for the general case of knowledge update.

Lemma 1 *Let $M = (w, W), M_1 = (w_1, W_1)$ and $M_2 = (w_2, W_2)$ be three k-models.*

1. *Deciding whether $KM \setminus KM_2 \neq \emptyset$ and $KM_2 \setminus KM \neq \emptyset$ has time complexity $\mathcal{O}(|W| \times |W_2|)$.*
2. *Deciding whether $KM \setminus KM_1 \subseteq KM \setminus KM_2$ has time complexity $\mathcal{O}(|W_1| \times (|W| + |W_2|))$.*
3. *Deciding whether $KM_1 \setminus KM \subseteq KM_2 \setminus KM$ is in co-NP.*

Proof: Result 1 is equivalently to deciding whether $W \not\subseteq W_2$ and $W_2 \not\subseteq W$ (proper set inclusion). Obviously, this can be verified in $\mathcal{O}(|W| \times |W_2|)$ time.

Now we prove Result 2. From set inclusion and intersection properties, it is easy to see that $KM \setminus KM_1 \subseteq KM \setminus KM_2$ iff $KM_2 \cap KM \subseteq KM_1 \cap KM$. Then from Proposition 1 (Results 2 and 4) of (Baral & Zhang 2001), it follows that $KM_2 \cap KM \subseteq KM_1 \cap KM$ iff $W_1 \cup W \subseteq W_2 \cup W$. Obviously, checking whether $W_1 \cup W \subseteq W_2 \cup W$ can be done in $\mathcal{O}(|W_1| \times (|W| + |W_2|))$ time.

Finally we prove Result 3. Similarly, from set inclusion and union properties, we have that $KM_1 \setminus KM \subseteq KM_2 \setminus KM$ iff $KM_1 \cup KM \subseteq KM_2 \cup KM$ iff $KM_1 \subseteq KM_2 \cup KM$. Now we show that checking whether $KM_1 \subseteq KM_2 \cup KM$ is in co-NP. To show $KM_1 \not\subseteq KM_2 \cup KM$ we need to find some ϕ such that $\phi \in KM_1$ but $\phi \notin KM_2 \cup KM$. A guess of such a formula ϕ can be done in polynomial time. From the definition of KM_1, we know that $\phi \in KM_1$ iff for each world $w' \in W_1$, $w' \models \phi$. Clearly, this can be verified in polynomial time. Similarly, to check whether $\phi \notin KM_2 \cup KM$, we only need to find a world w'' from W_2 and a world w from W such that $w'' \not\models \phi$ and $w \not\models \phi$. Also, this can be done in polynomial time. ■

Lemma 2 *Let M, M_1 and M_2 be three k-models. Deciding whether $M_1 \leq_M M_2$ is in co-NP.*

Proof: According to Definition 2, if $w_1 \neq w_2$, then $M_1 \leq_M M_2$ iff $(w_1 \setminus w \cup w \setminus w_1) \subseteq (w_2 \setminus w \cup w \setminus w_2)$. Clearly, this can be verified in polynomial time. If $w_1 = w_2$, then we need to check the following conditions: (i) If $W \subseteq W_1$, then $M_1 \leq_M M_2$ iff condition (a) or (b) in Definition 2 is satisfied. From Lemma 1, we know that deciding whether (a) and (b) to be true can be done in polynomial time. (ii) If $W_1 \subseteq W$, then $M_1 \leq_M M_2$ iff condition (a) or (c) in Definition 2 is satisfied. From Lemma 1, deciding whether (c) to be true is in co-NP. (iii) If $W \not\subseteq W_1$ and $W_1 \not\subseteq W$, then $M_1 \leq_M M_2$ iff conditions (b) and (c) should be satisfied. Again, deciding whether condition (c) to be true is in co-NP. So, the problem is in co-NP. ■

Lemma 3 *Let M, M' be two k-models and μ a S5 formula. Deciding whether $M' \in Res(M, \mu)$ is in Π_2^P.*

Proof: According to Theorem 1, if $M' \notin Res(M, \mu)$, there must exist another k-model M'' such that $M'' <_M M'$. A guess of a k-model M'' can be done in polynomial time. From Lemma 2, deciding whether $M'' \leq_M M'$ is in co-NP. Since $M'' <_M M'$ iff $M'' \leq_M M'$ and $M' \not\leq_M M''$, this follows that checking whether $M'' <_M M'$ can be polynomially decided with two queries to a co-NP oracle. So the problem is in Π_2^P (co-Σ_2^P). ■

Theorem 2 *Model checking for knowledge update is Σ_3^P-complete.*

Proof: Membership proof. From Definition 4, $M \in Mod(T \circ \mu)$ iff for some $M' \in Mod(T)$, $M \in Res(M', \mu)$. A guess of M' and check whether $M' \in Mod(T)$, i.e. $M' \models T$, can be achieved in polynomial time. According to Lemma 3, deciding whether $M \in Res(M', \mu)$ can be solved with one call to a co-Σ_2^P oracle. So the problem is in Σ_3^P.

Hardness proof. We prove the hardness by giving a polynomial transformation from deciding the validity of $\exists X \forall Y \exists Z E$, where E is a Boolean expression using propositional atoms over $X \cup Y \cup Z$. We construct formulas T, μ and a k-model M over propositional atoms $X \cup Y \cup Z \cup \hat{X} \cup \hat{Y} \cup \hat{Z} \cup \{a, b\}$, where $|\hat{X}| = |X|$, $|\hat{Y}| = |Y|$ and $|\hat{Z}| = |Z|$, and any two sets among X, Y, Z, $\hat{X}$, $\hat{Y}$, $\hat{Z}$, and $\{a, b\}$ are disjoint.

$$
\begin{aligned}
T &= K(X \equiv \hat{X}) \wedge Y \wedge \neg\hat{Y} \wedge K(Z \equiv \hat{Z}) \wedge a \wedge b,\\
\mu &= [K\neg X \wedge K\hat{X} \wedge K\neg Y \wedge K\neg Z \wedge K\hat{Z} \wedge\\
&\quad K(a \wedge \neg b)] \vee\\
&\quad [K(X \equiv \hat{X}) \wedge K\neg\hat{Y} \wedge K(Z \equiv \hat{Z}) \wedge\\
&\quad K(a \equiv b) \wedge K(b \equiv \neg E)],\\
M &= (\{w\}, w), \text{ where } w = \hat{X} \cup \hat{Z} \cup \{a\}.
\end{aligned}
$$

For simplicity, we denote $\mu = \gamma_1 \vee \gamma_2$, where

$$
\begin{aligned}
\gamma_1 &= K\neg X \wedge K\hat{X} \wedge K\neg Y \wedge K\neg Z \wedge K\hat{Z} \wedge\\
&\quad K(a \wedge \neg b), \text{ and}\\
\gamma_2 &= K(X \equiv \hat{X}) \wedge K\neg\hat{Y} \wedge K(Z \equiv \hat{Z}) \wedge\\
&\quad K(a \equiv b) \wedge K(b \equiv \neg E).
\end{aligned}
$$

Clearly, M is a k-model of μ since $M \models \gamma_1$. Now we show that $\exists X \forall Y \exists Z E$ is valid iff M is a k-model of $T \circ \mu$.

($\Rightarrow$) Suppose $\exists X \forall Y \exists Z E$ is valid. Let $X_1 \subseteq X$ be one of the valuations of X that makes $\forall Y \exists Z E$ valid. That is, for any $Y_1 \subseteq Y$, there exists some $Z_1 \subseteq Z$ such that $X_1 \cup Y_1 \cup Z_1 \models E$. Let $X_2 \subseteq X$ be one of the valuations of X that makes $\forall Y \exists Z E$ not valid[2]. That is, for some $Y_2 \subseteq Y$, no matter what $Z_2 \subseteq Z$ we choose E evaluates to false. This implies that γ_2 has two kinds of k-models as showed by M_1 and M_2 below:

$$
\begin{aligned}
&M_1 = (W_1, w_1),\\
&w_1 = X_1 \cup Y_1 \cup Z_1 \cup \hat{X}_1 \cup \hat{Z}_1{}^3,\\
&\text{where each world } w_1' \in W_1 \text{ has the same form}\\
&\text{of } w_1:\\
&w_1' = X_1' \cup Y_1' \cup Z_1' \cup \hat{X}_1' \cup \hat{Z}_1', \text{ where}\\
&X_1' \cup Y_1' \cup Z_1' \models E,
\end{aligned}
$$

$$
\begin{aligned}
&M_2 = (W_2, w_2),\\
&w_2 = X_2 \cup Y_2 \cup Z_2 \cup \hat{X}_2 \cup \hat{Z}_2 \cup \{a, b\},\\
&\text{where each world } w_2' \in W_2 \text{ has the same form}\\
&\text{of } w_2:\\
&w_2' = X_2' \cup Y_2' \cup Z_2' \cup \hat{X}_2' \cup \hat{Z}_2' \cup \{a, b\}, \text{ where}\\
&X_2' \cup Y_2' \cup Z_2' \not\models E.
\end{aligned}
$$

Note that $M_1 \models KE$, and $M_2 \models K\neg E$. We consider a k-model as follows:

$$
\begin{aligned}
&M^* = (\{w^*\}, w^*), \text{ where}\\
&w^* = X_1 \cup \hat{X}_1 \cup Y \cup Z_1 \cup \hat{Z}_1 \cup \{a, b\}.
\end{aligned}
$$

It is easy to verify that M^* is a k-model of T. Now we prove $M \in Res(M^*, \mu)$. According to Theorem 1, we only need to show $M \in Min(Mod(\mu), \leq_{M^*})$. We use notion $diff(w, w')$ to denote the symmetric difference between sets w and w', i.e. $w \setminus w' \cup w' \setminus w$.

[2] Note that if there does not exist such valuation of X, then γ_2 will only has one type of k-model M_1.

[3] Here we let $\hat{X}_1 = \{\hat{x}_i \mid x_i \in X_1\}$. This kind of denotation is adopted through out this paper whenever there is no explicit declaration.

$$diff(w^*, w) = X_1 \cup (\hat{X} \setminus \hat{X}_1) \cup Y \cup Z_1 \cup (\hat{Z} \setminus \hat{Z}_1) \cup \{b\},$$
$$diff(w^*, w_1) = (Y \setminus Y_1) \cup \{a, b\},$$
$$diff(w^*, w_2) = diff(X_1, X_2) \cup (Y \setminus Y_2) \cup diff(Z_1, Z_2) \cup diff(\hat{X}_1, \hat{X}_2) \cup diff(\hat{Z}_1, \hat{Z}_2).$$

Since $a \in diff(w^*, w_1)$ but $a \notin diff(w^*, w)$, $diff(w^*, w_1)$ is not a proper subset of $diff(w^*, w)$. On the other hand, as $\hat{X} \setminus \hat{X}_1 \subseteq diff(w^*, w)$ and $\hat{X}_1 \setminus \hat{X}_2 \subseteq diff(w^*, w_2)$[4], there exists some $\hat{x}_i \in \hat{X}_1$ such that $\hat{x}_i \in diff(w^*, w_2)$ but $\hat{x}_i \notin diff(w^*, w)$. So $diff(w^*, w_2)$ is not a proper subset of $diff(w^*, w)$ either. This follows that $M = (\{w\}, w) \in Min(Mod(\mu), \leq_{M^*})$, i.e. M is a k-model of $T \circ \mu$.

($\Leftarrow$) Now suppose $\exists X \forall Y \exists Z E$ is not valid. That is, $\forall X \exists Y \forall Z \neg E$ is valid. Consider any k-model of T:

$$M^* = (W^*, w^*), \text{ where}$$
$$w^* = X' \cup \hat{X}' \cup Y \cup Z' \cup \hat{Z}' \cup \{a, b\},$$

there always exists a k-model of μ given by

$$M' = (\{w'\}, w'), \text{ where}$$
$$w' = X' \cup \hat{X}' \cup Y'' \cup Z' \cup \hat{Z}' \cup \{a, b\}.$$

We can assume that $X' \subseteq X$ and $Y'' \subseteq Y$ is a valuation of Y that makes $\exists Z \neg E$ valid on X', i.e. $X' \cup Y'' \cup Z' \models \neg E$. It is easy to see that $M' \models \gamma_2$ and $M' \models K \neg E$. Now we show $diff(w^*, w') \subset diff(w^*, w)$. Indeed, we have

$$diff(w^*, w) = X' \cup (\hat{X} \setminus \hat{X}') \cup Y \cup Z' \cup (\hat{Z} \setminus \hat{Z}') \cup \{b\}, \text{ and}$$
$$diff(w^*, w') = (Y \setminus Y'') \subseteq Y \subset diff(w^*, w).$$

This concludes that M is not a k-model of $T \circ \mu$. ■

4 A Tractable Subclass - Knowledge Decreased Update

In this section, we identify a subclass of knowledge update problems for which model checking can be achieved in polynomial time. We first introduce a useful notation. Let α be a S5 formula and ϕ^α be an objective formula (i.e. no K occurs in it) occurring in α. We then say ϕ^α is an *objective sub-formula* of α. We denote the set of all objective sub-formulas of α as $Sub^o(\alpha)$. For instance, given $\alpha = Ka \vee K \neg b$, $Sub^o(\alpha) = \{a, b, \neg b\}$.

[4]Without loss of generality, we assume $\hat{X}_1 \setminus \hat{X}_2 \neq \emptyset$.

Definition 5 *Given S5 formulas T and μ, updating T with μ is called* knowledge decreased *if for any k-model $M' = (W', w')$ of $T \circ \mu$, there exists a k-model $M = (W, w)$ of T such that (i) $W \subseteq W'$, and $w = w'$; and (ii) for any $w^* \in W'$, $w^* \in W$ iff $w^* \models \phi^\mu$ or $w^* \models \neg \phi^\mu$ for some ϕ^μ in $Sub^o(\mu)$.*

From the above definition, it is easy to see that if an update is knowledge decreased, then the actual world of the agent's state will not change, and the agent's knowledge can only be decreased. Furthermore, the set of posible worlds in the agent's resulting state can be specifically computed from her previous state. We have the following important result on the model checking for knowledge decreased update.

Theorem 3 *Model checking for knowledge decreased update can be achieved in polynomial time.*

Proof: Given T, μ and a k-mode $M' = (W', w')$. Suppose $T \circ \mu$ be knowledge decreased. To check whether $M' \in Mod(T \circ \mu)$, we need to do the following things:

1. Check whether $M' \models \mu$,
2. Compute a subset W of W' such that for any $w^* \in W'$, $w^* \in W$ iff $w^* \models \phi^\mu$ or $w^* \models \neg \phi^\mu$ for some $\phi^\mu \in Sub^o(\mu)$,
3. Check whether $(W, w) \models T$.

Clearly, Steps 1 and 3 can be done in polynomial time. As $|Sub^o(\mu)| \leq |\mu|$, it follows that Step 2 can be also done in polynomial time. ■

Now it is worthwhile to examine some concrete forms of knowledge decreased update which have important applications in practical domains.

Ignorance update

We consider how an agent ignores a fact from her knowledge set. This procedure is so-called *ignorance update*, i.e. updating T with $\neg K\phi$. From Definition 1, it is easy to see that $T \circ \neg\phi \models \neg K\phi$. However, it should be noted that updating $T \circ \neg\phi$ can *not* be used to achieve $T \circ \neg K\phi$. Consider a k-model $M = (\{\{a, b\}, \{a\}\}, \{a, b\})$. Updating M with $\neg Ka$ we have a possible resulting k-model $M' = (\{\{a, b\}, \{a\}, \{b\}\}, \{a, b\})$, while updating M with $\neg a$ will lead to a possible result $M'' = (\{\{a, b\}, \{a\}, \{b\}\}, \{b\})$. Note that both M' and M'' entail $\neg Ka$, but $M' <_M M''$ according to Definition 2. In general, we have the following characteristic result for ignorance update.

Lemma 4 *(Baral & Zhang 2001) Given T and $\mu = \neg K\phi$ where ϕ is objective. $M' = (W', w')$ is a k-model of $T \circ \neg K\phi$ if and only if there exists a k-model $M = (W, w)$ of T such that*

(i) if $M \models K\phi$, then $w = w'$ and $W = W' \setminus \{w^*\}$, where $w^* \models \neg\phi$;

(ii) otherwise $w = w'$ and $W = W'$.

Theorem 4 *Ignorance update is knowledge decreased.*

Forgetting update

As another important type of knowledge update, we consider the update of T with $\mu = \neg K\phi \wedge \neg K\neg\phi$. This update can be thought of as the result of an agent *forgetting* her knowledge about the fact ϕ. We will refer to such an update as a *forgetting update*. Forgetting knowledge is an important issue in reasoning about knowledge dynamics, and forgetting update has a potential application to overcome the unrealistic assumption unlimited memory that has been adopted in many knowledge based system formalizations (Fagin *et al.* 1995). The following lemma shows that in order to forget ϕ from T, for each k-model of the current knowledge set, the agent only needs to expand the set of possible worlds of this model with exactly *one specific* world.

Lemma 5 *(Baral & Zhang 2001) Given T and $\mu = \neg K\phi \wedge \neg K\neg\phi$ where ϕ is objective. $M' = (W', w')$ is a k-model of $T \circ \mu$ if and only if there exists a k-model $M = (W, w)$ of T such that*

(i) if $M \models K\phi$, then $w = w'$ and $W = W' \setminus \{w^*\}$, where $w^* \models \neg\phi$;

(ii) if $M \models K\neg\phi$, then $w = w'$ and $W = W' \setminus \{w^*\}$, where $w^* \models \phi$;

(iii) otherwise $w = w'$ and $W = W'$.

Example 2 Suppose $T = Kb \wedge (Ka \vee K\neg a)$ represents the current knowledge of an agent. After executing a forgetting action the agent now would like to update her knowledge with $\mu = \neg Ka \wedge \neg K\neg a$. Let $w_0 = \{a, b\}, w_1 = \{b\}, w_2 = \{a\}, w_3 = \emptyset$. It is easy to see that $M_0 = (\{w_0\}, w_0)$ and $M_1 = (\{w_1\}, w_1)$ are the two k-models of T. Then using Proposition 5, we conclude that $M_0' = (\{w_0, w_1\}, w_0)$, $M_1' = (\{w_0, w_3\}, w_0)$, $M_2' = (\{w_1, w_0\}, w_1)$, and $M_3' = (\{w_1, w_2\}, w_1)$ are the four k-models of $T \circ \mu$. Note that $(\{w_0, w_2\}, w_0)$ cannot be a k-model of $T \circ \mu$ according to Lemma 5. ■

Theorem 5 *Forgetting update is knowledge decreased.*

Corollary 1 *Model checking for ignorance and forgetting updates can be achieved in polynomial time.*

5 Other Intractable Subclasses - Knowledge Increased and Gradual Updates

In this section, we address some other subclasses of knowledge update problems whose model checking complexity are intractable but still lower than the general case. Such investigation will be useful for us to design more optimal model checking algorithms for these subclasses of update problems.

5.1 Knowledge Increased Update

First, as a contrary case to the knowledge decreased update, the knowledge increased update is defined as follows.

Definition 6 *Given T and μ, updating T with μ is called* knowledge increased *if for any k-model $M' = (W', w')$ of $T \circ \mu$, there exists a k-model $M = (W, w)$ of T such that (i) $W' \subset W$, and $w = w'$; and (ii) for any $w^* \in W$, $w^* \in W'$ iff $w^* \models \phi^\mu$ or $w^* \models \neg\phi^\mu$ for some ϕ^μ in $Sub^o(\mu)$.*

It is clear that if a knowledge increased update is performed to an agent's knowledge set, it only increases the agent's knowledge and does not change the agent's actual world. Unfortunately, different from the knowledge decreased update, the model checking problem for knowledge increased update is not tractable. Before we prove this result, we first consider two specific forms of knowledge update that are knowledge increased.

Gaining knowledge and sensing updates

Given T and $K\phi$, where ϕ is objective and $T \not\models \phi$. We call $T \circ K\phi$ is *gaining knowledge update*, because it is easy to observe that the purpose of updating T with $K\phi$ is only to let the agent *know* ϕ.

Another interesting form of knowledge update is so-called *sensing update*, that is, updating T with $K\phi \vee K\neg\phi$. It has been proved that sensing update is particularly useful in reasoning about sensing action and planning (Scherl & Levesque 1993; Son & Baral 2001).

Now we introduce a notion that will be useful for us to provide alternative characterizations on these two

kinds of knowledge updates. Let W be a set of worlds and $w \in W$ and ϕ an objective formula. By $W^{(w,\phi)}$, we denote the subset of W: $\{w' \mid w' \in W$ and $w' \models \phi$ iff $w \models \phi\}$.

Lemma 6 *(Baral & Zhang 2001) Given T and $K\phi$ where ϕ is objective and $T \models \phi$. Then $M' = (W', w')$ is a k-model of $T \circ K\phi$ if and only if there exists a k-model $M = (W, w)$ of T such that $w = w'$ and $W' = W^{(w,\phi)}$.*

Lemma 7 *(Baral & Zhang 2001) Given T and $K\phi \vee K\neg\phi$ where ϕ is objective and $T \models \phi$. Then $M' = (W', w')$ is a k-model of $T \circ K\phi$ if and only if there exists a k-model $M = (W, w)$ of T such that $w = w'$ and $W' = W^{(w,\phi)}$, or $w = w'$ and $W' = W^{(w,\neg\phi)}$*

Theorem 6 *Gaining knowledge and sensing updates are knowledge increased.*

Theorem 7 *Model checking for knowledge increased update is NP-complete.*

Proof: Membership proof. Given T, μ and $M' = (W', w')$. To deciding whether $M' \in Mod(T \circ \mu)$, we only need to show that for some $M \in Mod(T)$, $M' \in Res(M, \mu)$. A guess of $M = (W, w)$ and verifying $M \models T$ can be done in polynomial time. Since $T \circ \mu$ is knowledge increased, to decide $M' \in Res(M, \mu)$, we only need to check: (1) $w = w'$, and (2) for any $w^* \in W$, $w^* \in W'$ iff $w^* \models \phi^\mu$ or $w^* \models \neg\phi^\mu$ for some $\phi^\mu \in Sub^o(\mu)$. Obviously, both (1) and (2) can be checked in polynomial time. So the problem is in NP.

Hardness proof. The hardness is proved by transforming the NP-complete SAT problem to a gaining knowledge update that has been showed to be knowledge increased. Let E be a CNF on the set of propositional atoms X. We construct formulas T, μ and a k-model M' over two disjoint sets X and $\hat{X}$ where $|X| = |\hat{X}|$.

$T = (X \equiv \hat{X}) \wedge \neg K(X \equiv \hat{X})$,
$\mu \equiv K(X \equiv \hat{X} \vee \neg E)$, and
$M' = (W', w')$, where
$W' = \{w'\}$, $w' = X \cup \hat{X}$.

Clearly, $M \models \mu$. We will show that E is satisfiable iff $M' \in Mod(T \circ \mu)$. Note that since $T \models X \equiv \hat{X} \vee \neg E$ and $\mu = K(X \equiv \hat{X} \vee \neg E)$, $T \circ \mu$ is a gaining knowledge update that is knowledge increased according to Theorem 6.

($\Rightarrow$) Suppose E is satisfiable. Let $X_1 \subseteq X$ such that $X_1 \models E$. We specify a k-model as follows:

$M^* = (W^*, w^*)$,
$W^* = \{w^*, w''\}$,
$w^* = w' = X \cup \hat{X}$, and
$w'' = X_1 \cup \hat{X}_1$, where $\hat{X}_1 = \{\hat{x}_i \mid \hat{x}_i \in \hat{X}$ and $x_i \notin X_1\}$.

Since $w'' \not\models X \equiv \hat{X}$, it is easy to see that $M^* \models \neg K(X \equiv \hat{X})$. Therefore, M^* is a k-model of T. On the other hand, since $w'' \models E$ and $w'' \not\models X \equiv \hat{X}$, it follows that $W' = \{w'\} = \{w^*\} = W^{*(w^*,\phi)}$, where $\phi = (X \equiv \hat{X}) \vee \neg E$. From Lemma 6, $M' \in Res(M^*, \mu)$, so $M' \in Mod(T \circ \mu)$.

($\Leftarrow$) Now suppose E is not satisfiable. That is, for any $X_1 \subseteq X$, $X_1 \models \neg E$. Then from Lemma 6, for any k-model of T of the form $M = (W, w)$, where $w \neq w'$, $M' \notin Res(M, \mu)$. We consider k-models of T of the form $M = (W, w)$ where $w = w'$ (note $w' \in W$). Without loss of generality, we assume that there is one world $w^* \in W$ such that $w^* \not\models X \equiv \hat{X}$, otherwise $M \models K(X \equiv \hat{X})$ and M cannot be a k-model of T. On the other hand, since E is not satisfiable, $\neg E$ must be true in each world in W. So $M \models K\neg E$ and hence $M \models K(X \equiv \hat{X} \vee \neg E)$. This implies that $W' \neq W^{(w,\phi)}$, where $\phi = (X \equiv \hat{X}) \vee \neg E$. So M' is not a k-model of $T \circ \mu$. ■

5.2 Knowledge Gradual Update

Now we consider a more general subclass of knowledge update problems that subsume both knowledge decreased and increased updates. Given T and μ, we say the update of T with μ is *knowledge gradual* if for any k-model $M' = (W', w')$ of $T \circ \mu$, there exists a k-model $M = (W, w)$ of T such that either $W \subseteq W'$ or $W' \subset W$. Note that, after performing a knowledge gradual update, the agent's knowledge may be decreased or increased (or without change), and the agent's actual world may be changed as well.

Example 3 Let $T = a \wedge \neg Ka$ and $\mu = K\neg a$. Obviously, T has a unique k-model $M = (\{\{a\}, \emptyset\}, \{a\})$. Then updating M with μ generates a unique k-model of $T \circ \mu$: $M' = (\{\emptyset\}, \emptyset)$. Obviously, M' has increased knowledge from M and the actual world of M' is also different from M's. ■

Lemma 8 *Let $M = (W, w)$, $M_1 = (W_1, w_1)$ and $M_2 = (W_2, w_2)$ be three k-models where $W \subseteq W_2$ or $W_2 \subseteq W$. Deciding whether $M_1 \leq_M M_2$ can be done in polynomial time.*

Proof: Obviously, if $w_1 \neq w_2$, then deciding whether $M_1 \leq_M M_2$ is in polynomial time according to condition 1 in Definition 2. Now suppose $w_1 = w_2$. If $W \subseteq W_2$, then from Definition 2, it is easy to see that

$M_1 \leq_M M_2$ iff $W \subseteq W_1 \subseteq W_2$. To see why this is the case, first consider $M_1 \leq_M M_2$. Since $W \subseteq W_2$, it is obviously that $W \not\subset W1$ and $W_1 \not\subset W$, otherwise $M_2 <_M M_1$. It cannot be $W_1 \subset W$ either since according to condition (ii) in Definition, condition (c) is not satisfied. So it must be the case that $W \subseteq W_1$ and condition (b) is satisfied. This implies that $W \subseteq W_1 \subseteq W_2$. On the ther hand, suppose $W \subseteq W_1 \subseteq W_2$. From condition (i) in Definition, this directly follows that $M_1 \leq_M M_2$.

Clearly, under the condition that $W \subseteq W_2$, checking whether $W \subseteq W_1 \subseteq W_2$ can be done in polynomial time.

Now consider the case that $W_2 \subseteq W$. We prove that under the condition $W_2 \subseteq W$, $M_1 \leq_M M_2$ iff $W_2 \subseteq W_1 \subseteq W$, from which it is clear that checking whether $M_1 \leq_M M_2$ can be verified in polynomial time. Obviously, if $W_2 \subseteq W_1 \subseteq W$, it directly follows that $M_1 \leq_M M_2$ from condition (ii) in Definition 2. Consider the other direction. If $M_1 \leq_M M_2$, then we have $W_1 \not\subset W$ and $W \not\subset W_1$ due to the fact $W_2 \subseteq W$, otherwise we will have $M_2 <_M M_1$. It is also impossible that $W \subseteq W_1$, since under this condition we can only use condition (b) to verify if $M_1 \leq_M M_2$ and this follows that (b) is not satisfied. So it must be the case $W_1 \subseteq W$. As $W_2 \subseteq W$, condition (c) must be satisfied, which implies $KM_1 \setminus KM \subseteq KM_2 \setminus KM$. From simple set inclusion and union properties, we have $KM_1 \setminus KM \subseteq KM_2 \setminus KM$ iff $KM_1 \cup KM \subseteq KM_2 \cup KM$. Since $W_1 \subseteq W$ and $W_2 \subseteq W$, we have $KM \subseteq KM_1$ and $KM \subseteq KM_2$. So $KM_1 \cup KM \subseteq KM_2 \cup KM$ iff $KM_1 \subseteq KM_2$. From Proposition 1 in (Baral & Zhang 2001), we have $KM_1 \subseteq KM_2$ iff $W_2 \subseteq W_1$. This completes our proof. ■

Lemma 9 *Let μ be a S5 formula and $M = (W, w)$, $M' = (W', w')$ two k-models where $W \subseteq W'$ or $W' \subseteq W$. Deciding whether $M' \in Res(M, \mu)$ is in co-NP.*

Proof: If $M' \notin Res(M, \mu)$, then there must exists a k-model M'' of μ such that $M'' \leq_M M'$ and $M' \not\leq_M M''$. Obviously, a guess of M'' and verifying $M'' \models \mu$ can be done in polynomial time. From Lemma 8, deciding whether $M'' \leq_M M'$ and $M' \not\leq_M M''$ can be done in polynomial time. So the original problem is in co-NP. ■

Now we have the following main result of model checking for gradual knowledge update.

Theorem 8 *Model checking for gradual knowledge update is Σ_2^P-complete.*

To prove Theorem 8, we need first to prove the following lemma.

Lemma 10 *Let $X, Y, \hat{X}, \hat{Y}$ be sets of propositional atoms and a be a propositional atom, where $|X| = |\hat{X}|$, $|Y| = |\hat{Y}|$ and any two sets of $X, Y, \hat{X}, \hat{Y}$ and $\{a\}$ are disjoint. Suppose ϕ is an objective formula formed based on $X \cup Y$. Let T and μ be the following two S5 formulas respectively:*

$$
\begin{array}{l}
T = \gamma_1 \vee \gamma_2, \text{ where} \\
\gamma_1 = (X \equiv \hat{X}) \equiv \phi \equiv a \wedge (Y \wedge \neg\hat{Y}) \wedge \\
\qquad \neg K \neg (a \wedge X \wedge \neg\hat{X} \wedge Y \wedge \hat{Y}), \\
\gamma_2 = ((X \wedge \neg\hat{X}) \equiv \neg\phi \equiv \neg a) \wedge \hat{Y}, \text{ and} \\
\mu = K(a \wedge X \wedge \neg\hat{X} \wedge Y \wedge \hat{Y}) \vee \\
\qquad K(\neg a \wedge X \wedge \neg\hat{X} \wedge (\bigvee \neg Y) \wedge \hat{Y}) \vee \\
\qquad K(\neg a \wedge X \wedge \neg\hat{X} \wedge Y \wedge \hat{Y})^5.
\end{array}
$$

Then $T \circ \mu$ is knowledge gradual.

Proof: To prove $T \circ \mu$ to be knowledge gradual, we need to show that for any k-model $M = (W, w) \in Mod(T)$, if $M' = (W', w') \in Res(M, \mu)$, then either $W \subseteq W'$ or $W' \subseteq W$. From the construction of T, it is easy to see that if $M \in Mod(T)$, then either $M \models \gamma_1$ or $M \models \gamma_2$, but $M \not\models \gamma_1 \wedge \gamma_2$. Based on this observation, our proof consists of two cases.

Case 1. Let $M = (W, w) \in Mod(\gamma_1)$. Since $M \models \gamma_1$, we have

$$w = X_1 \cup \hat{X}_1 \cup Y \cup \{a\}, \text{ where } X_1 \cup Y \models \phi \text{ for some } X_1 \subseteq X.$$

Furthermore, since $M \models \neg K \neg (a \wedge X \wedge \neg\hat{X} \wedge Y \wedge \hat{Y})$, there exists a world $w^* \in W$ such that

$$w^* = X \cup Y \cup \hat{Y} \cup \{a\}.$$

Now we specify a k-model of μ as follows:

$$M^* = (\{w^*\}, w^*).$$

It is easy to see that $M^* \models K(a \wedge X \wedge \neg\hat{X} \wedge Y \wedge \hat{Y})$. So $M^* \models \mu$. Furthermore, M^* is the unique k-model of $K(a \wedge X \wedge \neg\hat{X} \wedge Y \wedge \hat{Y})$. We prove M^* is the unique k-model in $Res(M, \mu)$.

Note $Diff(w, w^*) = (X \setminus X_1) \cup \hat{X}_1 \cup \hat{Y}$. Besides M^*, μ has other two types of k-models:

[5] Recall that $\bigvee \neg Y = \bigvee_{y_i \in Y} \neg y_i$.

$M_1 = (W_1, w_1)$, where $w_1 = X \cup Y_1 \cup \hat{Y}$, where $Y_1 \subset Y$ $(Y_1 \neq Y)$, and
$M_2 = (W_2, w_2)$, where $w_2 = X \cup Y \cup \hat{Y}$.

Note that

$M_1 \models K(\neg a \wedge X \wedge \neg\hat{X} \wedge (\bigvee \neg Y) \wedge \hat{Y})$, and
$M_2 \models K(\neg a \wedge X \wedge \neg\hat{X} \wedge Y \wedge \hat{Y})$.

Consider

$Diff(w, w_1) = (X \setminus X_1) \cup \hat{X_1} \cup (Y \setminus Y_1) \cup \hat{Y} \cup \{a\}$,
$Diff(w, w_2) = (X \setminus X_1) \cup \hat{X_1} \cup \hat{Y} \cup \{a\}$.

Clearly, we have

$Diff(w, w^*) \subset Diff(w, w_1)$, and
$Diff(w, w^*) \subset Diff(w, w_1)$.

So $Res(M, \mu) = \{M^*\}$. Also observe that $M^* = (\{w^*\}, w^*)$, $\{w^*\} \subset W$.

Case 2. Let $M = (W, w) \in Mod(\gamma_2)$. We have

$w = X \cup Y_1 \cup \hat{Y}$, where $X \cup Y_1 \models \neg\phi$ for some $Y_1 \subseteq Y$.

If $Y_1 \neq Y$, then we have

$$w \models \neg a \wedge X \wedge \neg\hat{X} \wedge (\bigvee \neg Y) \wedge \hat{Y},$$

this implies that there exists a subset W_1 of W where for each $w_i \in W_1$,

$$w_i \models (\neg a \wedge X \wedge \neg\hat{X} \wedge (\bigvee \neg Y) \wedge \hat{Y}).$$

By specifying W_1 to be the maximal such subset of W, it is easy to note that $M_1 = (W_1, w)$ is a k-model in $Res(M, \mu)$.

On the other hand, if $Y_1 = Y$, then we have

$$w \models (\neg a \wedge X \wedge \neg\hat{X} \wedge Y \wedge \hat{Y}),$$

this implies that there exists a subset W_2 of W where for each $w_i \in W_2$,

$$w_i \models (\neg a \wedge X \wedge \neg\hat{X} \wedge Y \wedge \hat{Y}).$$

Similarly, by specifying W_2 to be the maximal such subset of W, it is easy to note that $M_2 = (W_2, w)$ is a k-model in $Res(M, \mu)$.

Since in both cases, we have $W_1 \subseteq W$ and $W_2 \subseteq W$, this follows that for any k-model M of T where $M \models \gamma_2$, every resulting k-model after updating M with μ only increases the knowledge from M. ■

Proof: (Proof of Theorem 8) Membership proof. Given T, μ and a k-model M where $T \diamond \mu$ is knowledge gradual. $M \in Mod(T \diamond \mu)$ iff for some $M' \in Mod(T)$ such that $M \in Res(M', \mu)$. Obviously, a guess of M' and verifying $M' \models T$ can be done in polynomial time. From Lemma 9, deciding whether $M \in Res(M', \mu)$ can be solved with one call to a co-NP oracle. So the problem is in Σ_2^P.

Hardness proof. This part is based on a variation of the proof of Lemma 10. We prove the hardness by giving a polynomial transformation from deciding the validity of $\exists X \forall Y E$, where E is a Boolean expression using propositional atoms over $X \cup Y$. We construct T, μ and a k-model M^* over propositional atoms $X \cup Y \cup \hat{X} \cup \hat{Y} \cup \{a\}$, where $|\hat{X}| = |X|$ and $|\hat{Y}| = |Y|$, and any two sets among $X, Y, \hat{X}, \hat{Y}$ and $\{a\}$ are disjoint.

$T = \gamma_1 \vee \gamma_2$, where
$\gamma_1 = (X \equiv \hat{X}) \equiv E \equiv a \wedge (Y \wedge \neg\hat{Y}) \wedge$
$\quad \neg K \neg (a \wedge X \wedge \neg\hat{X} \wedge Y \wedge \hat{Y})$,
$\gamma_2 = ((X \wedge \neg\hat{X}) \equiv \neg E \equiv \neg a) \wedge \hat{Y}$,
$\mu = K(a \wedge X \wedge \neg\hat{X} \wedge Y \wedge \hat{Y}) \vee$
$\quad K(\neg a \wedge X \wedge \neg\hat{X} \wedge (\bigvee \neg Y) \wedge \hat{Y}) \vee$
$\quad K(\neg a \wedge X \wedge \neg\hat{X} \wedge Y \wedge \hat{Y})$,
$M^* = (W^*, w^*)$, where
$W^* = \{w^*\}$, $w^* = X \cup Y \cup \hat{Y} \cup \{a\}$.

Note that

$$M^* \models K(a \wedge X \wedge \neg\hat{X} \wedge Y \wedge \hat{Y}).$$

So M^* is a k-model of μ. Furthermore, it is the unique k-model of $K(a \wedge X \wedge \neg\hat{X} \wedge Y \wedge \hat{Y})$. From Lemma 10, we know that $T \diamond \mu$ is knowledge gradual. Now we will show that M^* is a k-model of $T \diamond \mu$ if and only if $\exists X \forall Y E$ is valid.

$(\Rightarrow)$ Suppose $\exists X \forall Y E$ is valid. Then for some $X_1 \subseteq X$, $X_1 \cup Y \models E$. We specify a k-model of γ_1 as follows:

$$M = (W, w), \text{ where } w = X_1 \cup \hat{X_1} \cup Y \cup \{a\}.$$

Since $M \models \neg K \neg (a \wedge X \wedge \neg\hat{X} \wedge Y \wedge \hat{Y})$, it is clear that the world w^* must be in W, i.e. $w^* \in W$. With the same justification as described in the proof of Lemma 10, we conclude that M^* is a k-model of updating M with μ.

($\Leftarrow$) Suppose $\exists X \forall Y E$ is not valid. That is, $\forall X \exists Y \neg E$ is valid. Then $X \cup Y_1 \models \neg E$ for some $Y_1 \subseteq Y$. In this case, T has the following type of k-models:

$$M = (W, w), \text{ where } w = X \cup Y_1 \cup \hat{Y}.$$

Note that $w \models ((X \wedge \neg \hat{X} \equiv \neg E \equiv \neg a) \wedge \hat{Y})$. That is, M is a k-model of γ_2.

If $Y_1 \neq Y$, then we have

$$w \models (\neg a \wedge X \wedge \neg \hat{X} \wedge (\bigvee \neg Y) \wedge \hat{Y}).$$

We now specify a k-model of μ as follows: $M_1 = (W_1, w_1)$, where $w_1 = w$ and W_1 is the maximal subset of W such that for each $w_i \in W_1$,

$$w_i \models (\neg a \wedge X \wedge \neg \hat{X} \wedge (\bigvee \neg Y) \wedge \hat{Y}).$$

Since

$$Diff(w, w^*) = (X \setminus X_1) \cup \hat{X_1} \cup \hat{Y}, \text{ and}$$
$$Diff(w, w_1) = \emptyset \subset Diff(w, w^*),$$

M^* is not a k-model in $Res(M, \mu)$.

If $Y_1 = Y$, then we have

$$w \models (\neg a \wedge X \wedge \neg \hat{X} \wedge Y \wedge \hat{Y}).$$

Again, we can specify a k-model of μ as follows: $M_2 = (W_2, w_2)$, where $w_2 = w$ and W_2 is maximal subset of W such that for each $w_i \in W_2$,

$$w_i \models (\neg a \wedge X \wedge \neg \hat{X} \wedge Y \wedge \hat{Y}).$$

Since $Diff(w, w_2) = \emptyset \subset Diff(w, w^*)$, M^* is not a k-model in $Res(M, \mu)$ in this case either.

Finally, suppose for some $X_1 \subseteq X$ and $Y_1 \subseteq Y$, E is evaluated to be true on $X_1 \cup Y_1$, i.e. $X_1 \cup Y_1 \models E$. Without loss of generality, we can assume $Y_1 \neq Y$[6]. This implies that γ_1 does not have a k-model under this situation. Therefore, if $\exists X \forall Y E$ is not valid, all k-models of T must be k-models of γ_2. ■

[6]Note that this assumption is always feasible. For instance, if $X_1 \cup Y \models E$, we can expand Y to be Y' by adding a new atom y' into Y to make $Y \neq Y'$, i.e. $Y' = Y \cup \{y'\}$, and modify E to be $E' = E \wedge \neg y'$ such that $X_1 \cup Y \models E'$ but $X_1 \cup Y' \not\models E'$.

6 Conclusion

The notions of belief revision and update have been well-studied in the literature (Gardenfors 1988; Katsuno & Mendelzon 1991; Winslett 1988; Zhang & Foo 2000) and the computational complexity of model checking and performing such revision/update and its variations were well understood (Eiter & Gottlob 1992; Liberatore 2000; Liberatore & Schaerf 2001; Zhang 2001). On the other hand, although the issue of reasoning about knowledge dynamics has been investigated by some researchers from different perspectives, e.g. (Fagin *et al.* 1995; Groeneveld 1995; Lomuscio & Ryan 1998), the formal account of modeling knowledge change from update viewpoint is far from clear. In (Baral & Zhang 2001) we introduced the notion of knowledge update, defined its formal semantics and related it to the impact of sensing actions on the knowledge of an agent, and to actions corresponding to forgetting and ignorance. We also argued why such a new notion of knowledge update is necessary for the above and why the earlier notions of belief revision/update are not sufficient.

In this paper we undertook a further study about the complexity issue of knowledge update. In particular, we analyzed the complexity of model checking for knowledge update in the general case and in special cases. We identify special subcases where the model checking is either tractable or its complexity is lower than the general case (see the summary in the following table). We expect that these results would be useful for designing more optimal model checking algorithms in the implementation of knowledge update (Baral & Zhang 2002).

Update Type		Complexity
General case		Σ_3^P-complete
Knowledge decreased		
ignorance	forgetting	P
Knowledge increased		
gaining knowledge	sensing	NP-complete
Knowledge gradual		Σ_2^P-complete

References

Baral, C., and Zhang, Y. 2001. On the semantics of knowledge update. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, 97–102.

Baral, C., and Zhang, Y. 2002. Knowledge update: Semantics and complexity. *Manuscript.*

Eiter, T., and Gottlob, G. 1992. On the complexity of

propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence* 57:227–270.

Fagin, R.; Halpern, J.; Moses, Y.; and Vardi, M. 1995. *Reasoning about Knowledge.* MIT Press.

Gardenfors, P. 1988. *Knowledge in Flux.* Cambridge, Massachusetts: MIT Press.

Garey, M., and Johnson, D. 1979. *Computers and Intractability.* Freeman.

Groeneveld, W. 1995. *Logical Investigations into Dynamic Semantics.* PhD Thesis, Universiteit van Amsterdam.

Katsuno, H., and Mendelzon, A. 1991. On the difference between updating a knowledge base and revising it. In *Proceedings of KR-91*, 387–394.

Liberatore, P., and Schaerf, M. 2001. Belief revision and update: complexity of model checking. *Journal of Computer and System Sciences* 62:43–72.

Liberatore, P. 2000. The complexity of belief update. *Artificial Intelligence* 119:141–190.

Lomuscio, A., and Ryan, M. 1998. Ideal agents sharing (some!) knowledge. In *Proceedings of ECAI-98*, 557–561.

Meyer, J.-J., and van der Hoek, W. 1995. Epistemic logics for computer science and artificial intelligence. In *Cambridge Tracts in Theoretical Computer Science, 41.* Cambridge University Press.

Scherl, R., and Levesque, H. 1993. The frame problem and knowledge producing actions. In *Proceedings of AAAI-93*, 689–695.

Son, T., and Baral, C. 2001. Formalizing sensing actions: a transition function based approach. *Artificial Intelligence* 125:19–91.

Winslett, M. 1988. Reasoning about action using a possible models approach. In *Proceedings of the 7th National Conference on AI (AAAI-88)*, 89–93.

Zhang, Y., and Foo, N. 2000. Updates with disjunctive information: From syntactical and semantical perspectives. *Computational Intelligence* 16:29–52.

Zhang, Y. 2001. The complexity of logic program updates. In *Proceedings of 14th Australian Joint Conference on Artificial Intelligence (AI2001)*, 630–643.

Merging and Integration I

Distance-based Merging: A General Framework and some Complexity Results

Sébastien Konieczny
IRIT
Université Paul Sabatier
31602 Toulouse Cedex - France
konieczny@irit.fr

Jérôme Lang
IRIT
Université Paul Sabatier
31602 Toulouse Cedex - France
lang@irit.fr

Pierre Marquis
CRIL / Université d'Artois
rue de l'Université - S.P. 16
62307 Lens - France
marquis@cril.univ-artois.fr

Abstract

We present in this paper a new framework for propositional merging. *Distance-based merging operators*, parameterized by a distance between interpretations and two aggregation functions, are introduced. Many distances and aggregation functions can be used and many merging operators already defined in the literature (including both model-based ones and syntax-based ones) can be recovered as specific distance-based operators. Both logical and complexity properties of distance-based merging operators are studied. An important result is that (under very weak assumptions) query entailment from merged bases is "only" at the first level of the polynomial hierarchy when any of our distance-based operators is used. As a by-product, complexity results for several existing merging operators are derived as well.

1 INTRODUCTION

Belief merging is an important issue of many AI fields (see [Bloch and Hunter, 2001] for a panorama of applications of data and knowledge fusion).

Although particular requirements can be asked for each application, several pieces of information are usually brought into play when propositional base merging is concerned. In the following:

- A *knowledge set* $E = \{K_1, \ldots, K_n\}$ is a finite multi-set of knowledge bases, where each *knowledge base* K_i represents the set of beliefs from source i. Each K_i is a propositional formula, or more generally, a finite set of propositional formulas $\varphi_{i,j}$ encoding the explicit beliefs from source i.
- Some *integrity constraints* IC encoded as a propositional formula. IC represents some common knowledge on which all sources agree (e.g. some physical constraints, norms, etc.).

The purpose of merging E is to characterize a formula (or a set of formulas) $\Delta_{IC}(E)$, considered as the overall knowledge from the m sources given the integrity constraints IC. Recently, several families of such merging operators have been defined and characterized in a logical way [Revesz, 1997; Lin and Mendelzon, 1999; Liberatore and Schaerf, 1998; Konieczny and Pino Pérez, 1999; Benferhat *et al.*, 2000]. Among them are the so-called *model-based* merging operators [Revesz, 1997; Lin and Mendelzon, 1999; Liberatore and Schaerf, 1998; Konieczny and Pino Pérez, 1999] where the models of $\Delta_{IC}(E)$ are defined as the models of IC which are preferred according to some criterion depending on E. Often, such preference information take the form of a total pre-order on interpretations, induced by a notion of distance $d(\omega, E)$ between an interpretation ω and the knowledge set E. $d(\omega, E)$ is typically defined by aggregating the distances $d(\omega, K_i)$ for every K_i. Usually, model-based merging operators takes only into account consistent knowledge bases K_i. Other merging operators are *syntax-based* ones [Baral *et al.*, 1991; Baral *et al.*, 1992; Konieczny, 2000]. They are based on the selection of some consistent subsets of $\bigcup_{i=1}^{m} K_i$. This renders possible to take into account inconsistent knowledge bases K_i and to incorporate some additional preference information into the merging process[1] but the price to be paid is to give some importance to the syntax of knowledge bases. Moreover, since they are based on the set-theoretic union $\bigcup_{i=1}^{m} K_i$ of the bases, such operators usually do not take into account the frequency

[1] Indeed, as in belief revision, giving some importance to the syntax of K_i is a way to specify (implicitly but in a cheap way w.r.t. representation) that explicit beliefs are preferred to implicit beliefs [Nebel, 1989; Hansson, 1998].

of each explicit pieces of belief into the merging process (the fact that $\varphi_{i,j}$ is believed in one source only or in the m sources under consideration is not considered relevant, which is often counter-intuitive[2]).

In this paper, a new framework for defining propositional merging operators is provided. A family of merging operators parametrized by a distance d between interpretations and two aggregation functions f and g is presented. These parameters are used to define a notion of distance between an interpretation and a knowledge set E in a two-step fashion. Like in existing model-based approaches to merging, the models of the merging of E given some integrity constraints IC are exactly the models of IC that are as close as possible to E with respect to the distance. Moreover, the first aggregation step enables to take into account the syntax of knowledge bases within the merging process. This allows to handle inconsistent ones in a satisfying way.

The contribution of this work is many fold. First, our framework is general enough to encompass almost all model-based merging operators as specific cases. In addition, despite the model-theoretic ground of our approach, several syntax-based merging operators provided so far in the literature can be captured as well. We show that, by imposing few conditions on the parameters, several logical properties that are expected when merging operators are considered, are already satisfied.

Another very strong feature offered by our framework is that query entailment from $\Delta_{IC}(E)$ is guaranteed to lay at the first level of the polynomial hierarchy provided that d, f and g can be computed in polynomial time. Accordingly, improving the generality of the model-based merging operators framework through an additional aggregation step does not result in a complexity shift.

We specifically focus on some simple families of distances and aggregation functions. By letting the parameters d, f and g vary in these respective sets, several merging operators are obtained; some of them were already known and are thus recovered as specific cases in our framework, and others are new operators. In any case, we investigate the logical properties and identify the complexity of each operator under consideration. As a by-product, the complexity of several model-based merging operators already pointed out so far is also identified.

The full proofs of the results given in this article can be found in [Konieczny *et al.*, 2001].

[2]See [Konieczny, 2000] for one step in that direction.

2 FORMAL PRELIMINARIES

We consider a propositional language $PROP_{PS}$ built up from a finite set PS of propositional symbols in the usual way. An interpretation is a total function from PS to $BOOL = \{0, 1\}$. The set of all interpretations is denoted $\mathcal{W}$. An interpretation ω is a model of a formula iff it makes it true in the usual classical truth functional way. Provided that φ is a formula from $PROP_{PS}$, $Mod(\varphi)$ denotes the set of models of φ, i.e., $Mod(\varphi) = \{\omega \in \mathcal{W} \mid \omega \models \varphi\}$.

A knowledge base K_i is said to be consistent iff the conjunction $\bigwedge K_i$ of its formulas is consistent. Similarly, a knowledge set E is said to be consistent iff the conjunction $\bigwedge E$ of its knowledge bases is consistent. Two knowledge bases K_1 and K_2 are said to be logically equivalent ($K_1 \equiv K_2$) iff $\bigwedge K_1 \equiv \bigwedge K_2$, and two knowledge sets E_1 and E_2 are said to be equivalent ($E_1 \equiv E_2$) iff there is a bijection between E_1 and E_2 such that each knowledge base of E_1 is logically equivalent to its image in E_2. $\sqcup$ denotes the multi-set union. For every knowledge set E and for every integer n, E^n denotes the multi-set obtained by "unioning" E with itself n times.

The complexity results we give in this paper refer to some complexity classes which we now briefly recall (see [Papadimitriou, 1994] for more details), especially the classes Δ_2^p and Θ_2^p [Eiter and Gottlob, 1992; Wagner, 1987] from the polynomial hierarchy PH, as well as the class BH_2 from the Boolean hierarchy (see [Papadimitriou, 1994]). Given a problem A, we denote by $\overline{\mathrm{A}}$ its complement. We assume the reader familiar with the classes P, NP et coNP and we now introduce the following three classes located at the first level of the polynomial hierarchy:

- BH_2 (also known as DP) is the class of all languages L such that $L = L_1 \cap L_2$, where L_1 is in NP and L_2 in coNP. The canonical BH_2-complete problem is SAT–UNSAT: given two propositional formulas φ and ψ, $\langle \varphi, \psi \rangle$ is in SAT–UNSAT if and only if φ is consistent and ψ is inconsistent.
- $\Delta_2^p = \mathsf{P}^{\mathsf{NP}}$ is the class of all languages that can be recognized in polynomial time by a Turing machine equipped with an NP oracle, where an NP oracle solves whatever instance of a problem NP in unit time.
- $\Theta_2^p = \Delta_2^p[\mathcal{O}(log\ n)]$ is the class of all languages that can be recognized in polynomial time by a Turing machine using a number of NP oracles bounded by a logarithmic function of the size of the input data.

Note that the following inclusions hold:
$\mathsf{NP} \cup \mathsf{coNP} \subseteq \mathsf{BH}_2 \subseteq \Theta_2^p \subseteq \Delta_2^p \subseteq \mathsf{PH}$.

3 DISTANCE-BASED MERGING

3.1 THE GENERAL FRAMEWORK

Defining a merging operator in our framework simply consists in setting three parameters: a distance d and two aggregation functions f and g. Let us first make precise what such notions mean in this paper:

Definition 1 (distances) *Let d be a total function from $\mathcal{W} \times \mathcal{W}$ to $\mathbb{N}$ s.t. (1) for every ω_1, $\omega_2 \in \mathcal{W}$, $d(\omega_1,\omega_2) = d(\omega_2,\omega_1)$ and (2) $d(\omega_1,\omega_2) = 0$ iff $\omega_1 = \omega_2$. Such a d is called a* distance *between interpretations*[3]*. d induces a distance between any interpretation ω and any formula φ given by $d(\omega,\varphi) = \min_{\omega' \models \varphi} d(\omega,\omega')$.*

Definition 2 (aggregation functions) *Let f be a total function associating a nonnegative integer to every finite tuple of nonnegative integers and s.t.*

- *f is non-decreasing in each argument*[4]*, and*
- *f satisfies (minimality) : for every n-uple $(x_1,\ldots,x_n)$ of nonnegative integers, $f(x_1,\ldots,x_n) = 0$ iff $x_1 = \ldots = x_n = 0$, and*
- *for every nonnegative integer x_1, $f(x_1) = x_1$.*

f is called an aggregation function[5].

We are now in position to define our distance-based merging operators:

Definition 3 (distance-based merging operators) *Let d be a distance between interpretations and f and g be two aggregation functions. For every knowledge set $E = \{K_1,\ldots,K_n\}$ and every integrity constraint IC, $\Delta_{IC}^{d,f,g}(E)$ is defined in a model-theoretical way by:*

$$Mod(\Delta_{IC}^{d,f,g}(E)) = \{\omega \in Mod(IC) \mid d(\omega,E) \text{ is minimal}\}$$

where

$$d(\omega,E) = g(d(\omega,K_1),\ldots,d(\omega,K_n))$$

and for every $K_i = \{\varphi_{i,1},\ldots,\varphi_{i,n_i}\}$

$$d(\omega,K_i) = f(d(\omega,\varphi_{i,1}),\ldots,d(\omega,\varphi_{i,n_i})).$$

Formulas appearing in a knowledge base K_i can have various possible meanings, for instance:

- *pieces of information provided by the source i*: when merging several beliefs stemming from different sources (sensors or experts, for example);
- *pieces of information pertaining to a criterion i*: when evaluating alternatives with respect to different criteria;
- *elementary goals expressed by the agent i*: when aggregating individual preferences in a group decision making context – see [Lafage and Lang, 2000]. In this case, the formulas $\varphi_{i,j}$ are no longer *beliefs* but *preferences* (which does not prevent us from using the same merging operators).

The reason why we use *two* (generally distinct) aggregation functions f and g is that both aggregation steps are of different nature. The first step is an *intra-source* aggregation: f aggregates scores w.r.t. the elementary (explicit) pieces of information contained in each K_i (it allows, in particular, to take inconsistent knowledge bases into account). The second step is an *inter-source* aggregation: g aggregates the "f-aggregated scores" pertaining to the different sources.

Interestingly, few conditions are imposed on d, f, and g. As we will see in the next section, many distances and aggregation functions can be used. Often, the aggregation functions f and g are required to be symmetric (i.e., no priority is given to some explicit beliefs in a knowledge base, and no priority is given to some knowledge bases in a knowledge set). However, this condition is not mandatory here and this is important when some preference information is available, especially when all sources i are not equally reliable. For instance, the *weighted sum* aggregation function can be used to give rise to (non-symmetric) merging operators.

Let us stress that, contrarily to usual model-based operators, our definition allows inconsistent knowledge bases to take (a non-trivial) part in the merging process.

Example 1 *Assume for example that we want to merge $E = \{K_1,K_2,K_3,K_4\}$ under the integrity constraints $IC = \top$, where*

[3] We slightly abuse words here, since d is only a pseudo-distance (triangular inequality is not required).

[4] I.e., if $x \leq y$, then $f(x_1,\ldots,x,\ldots,x_n) \leq f(x_1,\ldots,y,\ldots,x_n)$.

[5] The aggregation function f can take arbitrarily many arguments; more formally, a "function" f is a family $f = \{f_n \mid n \in \mathbb{N}\}$ of n-ary functions from $\mathbb{N}^n$ to $\mathbb{N}$. Slightly abusing notations, we write $f(x_1,\ldots,x_n)$ instead of $f_n(x_1,\ldots,x_n)$ since this can never be ambiguous.

- $K_1 = \{a, b, c, a \rightarrow \neg b\}$,
- $K_2 = \{a, b\}$,
- $K_3 = \{\neg a, \neg b\}$,
- $K_4 = \{a, a \rightarrow b\}$.

In this example, K_1 knows that c holds; since this piece of information is not involved in any contradiction, it can prove sensible to be confident in K_1 about the truth of c. Model-based merging operators can not handle this situation: inconsistent knowledge bases can not be taken into account. Thus, provided that the Hamming distance between interpretations is considered, the operator Δ^{Σ} [Revesz, 1997; Lin and Mendelzon, 1999; Konieczny and Pino Pérez, 1999] gives a merged base whose models are: $\{a, b, \neg c\}$ and $\{a, b, c\}$; the operator Δ^{Gmax} [Konieczny and Pino Pérez, 1999] gives a merged base whose models are: $\{\neg a, b, \neg c\}$, $\{\neg a, b, c\}$, $\{a, \neg b, \neg c\}$, and $\{a, \neg b, c\}$. In any of these two cases, nothing can be said about the truth of c in the merged base, which is counter-intuitive since no argument against it can be found in the input.

Syntax-based operators render possible the exploitation of inconsistent knowledge bases, but they do not care about the distribution of information. Consider the two standard syntax-based operators [Baral *et al.*, 1992], selecting the maximal subsets of $\bigcup_{i=1}^{m} K_i$ (one w.r.t. set inclusion and the other one w.r.t. cardinality). On the previous example, the first one returns a merged base equivalent to c and the second one to $c \wedge \neg a$. So, a is in the result for none of these two operators, whereas a holds in three over four input bases.

Our distance-based operators achieve a compromise between model-based operators and syntax-based operators, by taking into account the way information is distributed and by taking advantage of the information stemming from inconsistent knowledge bases. For instance, our operator $\Delta^{d_D, sum, sum}$ (cf. Section 3.2) gives a merged base whose single model is $\{a, b, c\}$, and $\Delta^{d_D, sum, lex}$ returns a merged base whose models are $\{\neg a, b, c\}$ and $\{a, \neg b, c\}$. So, with any of these two operators, we can deduce that c holds after the merging. Moreover, these operators exhibit typical merging behaviours. The first one is a majority operator: since three of four bases agree on a, a holds in the result. The second one is an arbitration operator; being more consensual, it gives that only one of a or b holds, to be as close as possible to each of the knowledge bases.

3.2 INSTANCIATING OUR FRAMEWORK

Let us now instantiate our framework and focus on some simple families of distances and aggregation functions.

Definition 4 (some distances) *Let ω_1, $\omega_2 \in \mathcal{W}$ be two interpretations.*

- *The* drastic distance d_D *is defined by*
$$d_D(\omega_1, \omega_2) = 0 \text{ if } \omega_1 = \omega_2, \quad 1 \text{ otherwise}$$
- *The* Hamming distance d_H *is defined by*
$$d_H(\omega_1, \omega_2) = |\{x \in PS \mid \omega_1(x) \neq \omega_2(x)\}|$$
- *Let q be a total function from PS to $\mathbb{N}^*$. The* weighted Hamming distance d_{H_q} *induced by q is defined by*
$$d_{H_q}(\omega_1, \omega_2) = \sum_{x \in PS \mid \omega_1(x) \neq \omega_2(x)} q(x)$$

These distances satisfy the requirements imposed in Definition 3.

The Hamming distance is the most usual distance considered in model-based merging[6]. It is very simple to express, but one has to keep in mind that it is very sensitive to the representation language of the problem (i.e., the choice of propositional symbols) and that numerous others distances can be used. Weighted Hamming distances are relevant when some propositional symbols are known as more important than others.

Definition 5 (some aggregation functions)

- *Let q be a total function from $\{1, \ldots, n\}$ to $\mathbb{N}^*$ s.t. $q(1) = 1$ whenever $n = 1$. The weighted sum WS_q induced by q is defined by $WS_q(e_1, \ldots, e_n) = \sum_{i=1}^{n} q(i) e_i$.*
- *Let q be a total function from $\{1, \ldots, n\}$ to $\mathbb{N}$ s.t. $q(1) = 1$ whenever $n = 1$, and $q(1) \neq 0$ in any case. The ordered weighted sum OWS_q induced by q is defined by $OWS_q(e_1, \ldots, e_n) = \sum_{i=1}^{n} q(i) e_{\sigma(i)}$ where σ is a permutation of $\{1 \ldots n\}$ s.t. $e_{\sigma(1)} \geq e_{\sigma(2)} \geq \ldots \geq e_{\sigma(n)}$.*

q is a *weight function*, that gives to each formula (resp. knowledge base) φ_i (resp. K_i) of index i its weight $q(i)$ denoting the formula (resp. knowledge base) reliability. With the slight difference that q is normalized (but without requiring that $q(1) = 1$ whenever $n = 1$), the latter family is well-known in multi-criteria decision

[6] In this context, it is also called *Dalal distance* [Dalal, 1988].

$$
\begin{array}{rcl}
\Delta^{d_D,max,max} & = & \top \\
\Delta^{d_D,max,sum}, \Delta^{d_D,max,lex}, \Delta^{d_H,max,sum} & = & a \wedge b \\
\Delta^{d_D,sum,max} & = & \neg b \\
\Delta^{d_D,sum,sum} & = & (\neg a \wedge \neg b) \vee (a \wedge b \wedge c) \\
\Delta^{d_D,sum,lex} & = & \neg a \wedge \neg b \\
\Delta^{d_H,sum,max}, \Delta^{d_H,sum,lex} & = & a \wedge \neg b \wedge c \\
\Delta^{d_H,max,max}, \Delta^{d_H,max,lex} & = & (\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge c) \\
\Delta^{d_H,sum,sum} & = & a \wedge c
\end{array}
$$

Figure 1: Example 2

making under the terminology "Ordered Weighted Averages" (OWAs) [Yager, 1998]. When $q(i) = 1$ for every $i \in 1 \ldots n$, WS_q is the usual sum (and OWS_q as well). When $q(1) = 1$ and $q(2) = \ldots = q(n) = 0$ then $OWS_q(e_1, \ldots, e_n) = max(e_1, \ldots, e_n)$.

For the second aggregation step g, it is relevant to consider the well-known *leximax* ordering which compares two vectors of scores by focusing on the largest scores of each vector, and in case of equality, on the second largest scores, and so on. For the sake of homogeneity, we reformulate the leximax ordering so as to compare aggregated scores rather than vectors of scores. This can be done thanks to a specific aggregation function OWS_q:

Definition 5.1 (leximax)
Let M be an upper bound of the scores $d(\omega, K_i)$[7], i.e., for any ω we have $d(\omega, K_i) < M$. Now, let $q(i) = M^{n-i}$ for all i. The rank order on vectors of scores induced by OWS_q is the leximax *ordering, abbreviated by lex*[8].

Using the *leximax* aggregation for the first aggregation step (f) would also be possible, but leads to rather lengthy technical tricks to be defined properly in case where the second aggregation function g is not purely ordinal (i.e., g different from *max* and *leximax*) and we ignore this possibility here (see the long version of the paper [Konieczny *et al.*, 2001]).

All these functions satisfy the requirements imposed in Definition 3; all of them are symmetric but *weighted sum* (except when q is uniform).

Many other possible choices for f and g can be found in the literature of multi-criteria decision making (and to a smaller extent in the literature of group decision theory). Noticeably, the usual aggregation functions used in these fields are all polynomially computable, which makes the following complexity results applicable when instantiating f and g with these functions.

Note that functions such as the purely utilitarian *sum* or *weighted sum* allow for compensations between scores (and lead to majority-like operators), while the egalitarian functions *max* and *lex* do not.

By letting the parameters d, f and g vary in these respective sets, several merging operators are obtained; some of them were already known and are thus recovered as specific cases in our framework, and others are new operators. Thus, $\Delta^{d_D,max,max}$ is the *basic* merging operator, giving $\bigwedge E \wedge IC$ if consistent and IC otherwise. $\Delta^{d_D,max,sum}$ is the *drastic* merging operator which amounts to select the models of IC satisfying the greatest number of knowledge bases from E. It is equivalent to the drastic majority operator as defined in [Konieczny, 2000] when working with deductively closed knowledge bases. $\Delta^{d_D,sum,sum}$ corresponds to the intersection operator of [Konieczny, 2000]. $\Delta^{d_D,WS_q,max}$ corresponds to an operator used in [Lafage and Lang, 2000] in a group decision context. When singleton knowledge bases are considered [9] – recall that in this case f is irrelevant – every $\Delta^{d_H,f,max}$ operator is a Δ^{Max} operator [Revesz, 1997], every $\Delta^{d_H,f,sum}$ operator is a Δ^{Σ} operator [Revesz, 1997; Lin and Mendelzon, 1999; Konieczny and Pino Pérez, 1999], and every $\Delta^{d_H,f,lex}$ operator is a Δ^{GMax} operator [Konieczny and Pino Pérez, 1999]. Still with singleton knowledge bases, taking $d = d_D$ and $f = WS_q$, Δ^{d_H,f,WS_q} is a penalty-based merging (where one minimizes the sum of the penalties $q(i)$ attached to the K_i's) [Pinkas, 1995], and taking $d = d_D$ and $f = WMAX_q$ (defined by $WMAX_q(x_1, \ldots, x_n) = \max_{i=1\ldots n} \min(q(i), x_i)$) we get[10] a possibilistic merging operator [Benferhat *et al.*, 2000].

[7] For instance, when $d = d_H$ and $f = \max$ we can choose $M = |PS| + 1$; when $d = d_H$ and $f = \sum$ we can choose $M = |PS|^2 + 1$.

[8] Namely, we have $OWS_q(e_1, \ldots, e_n) \geq OWS_q(e'_1, \ldots, e'_n)$ iff $(e_{\sigma(1)} > e'_{\sigma'(1)})$ or $(e_{\sigma(1)} = e'_{\sigma'(1)}$ and $e_{\sigma(2)} > e'_{\sigma'(2)})$ or etc.

[9] Or, equivalently, when each K_i is replaced by $\{\bigwedge K_i\}$ before merging.

[10] The scales used for scores are different but it is obvious to show that this difference has no impact.

Table 1: $\Delta^{d_H,sum,lex}$ Operator

	$a \wedge b \wedge c$	$a \rightarrow \neg b$	$a \wedge b$	$\neg a \wedge \neg b$	$\neg b$	a	$a \rightarrow b$	$\mathbf{K_1}$	$\mathbf{K_2}$	$\mathbf{K_3}$	$\mathbf{K_4}$	$\mathbf{E}$
$(0,0,0)$	3	0	2	0	0	1	0	3	2	0	1	3210
$(0,0,1)$	2	0	2	0	0	1	0	2	2	0	1	2210
$(0,1,0)$	2	0	1	1	1	1	0	2	1	2	1	2211
$(0,1,1)$	1	0	1	1	1	1	0	1	1	2	1	2111
$(1,0,0)$	2	0	1	1	0	0	1	2	1	1	1	2111
$(1,0,1)$	1	0	1	1	0	0	1	1	1	1	1	1111
$(1,1,0)$	1	1	0	2	1	0	0	2	0	3	0	3200
$(1,1,1)$	0	1	0	2	1	0	0	1	0	3	0	3100

We will now illustrate the behaviour of these different operators on an example.

Example 2 *Consider the following knowledge set* $E = \{K_1, K_2, K_3, K_4\}$ *that we want to merge under the integrity constraints* $IC = \top$.

- $K_1 = \{a \wedge b \wedge c, a \rightarrow \neg b\}$,
- $K_2 = \{a \wedge b\}$,
- $K_3 = \{\neg a \wedge \neg b, \neg b\}$,
- $K_4 = \{a, a \rightarrow b\}$.

The result of the merging of E *according to the different operators with* $d \in \{d_D, d_H\}$, $f \in \{max, sum\}$ *and* $g \in \{max, sum, lex\}$ *under no constraints (i.e.* $IC = \top$*) is indicated figure 1. See table 1 for an example of calculation with the* $\Delta^{d_H,sum,lex}$ *operator. In this table the interpretation* $(1,0,0)$ *for example is the one mapping* a *to true and* b *and* c *to false. The result of the merging* $\Delta_{\top}^{d_H,sum,lex}(E)$ *is the interpretation that is the closest to* E*, that is the one at a distance* 1111*, i.e. the one mapping* a *and* c *to true and* b *to false.*

The wide variety of obtained results show the degree of freedom given by this framework. This example illustrates several aspects of merging operators : the knowledge base K_1 is not consistent, but it is the only base that gives an information about c, so it can be sensible to take c as true in the result of the merging. K_3 is logically equivalent to $\neg a \wedge \neg b$, but replacing K_3 by this formula would lead to different results for merging. Syntax is relevant for distance-based merging operators since one has to consider that different formulae of a same base are distinct reasons to believe in a same information. Taking syntax into account is important from the point of view of representation of beliefs (or goals), but the operators can then "choose" to take or not this information into account. So the point in this framework is that, unlike classical model-based merging operators, the connector "," is not the same that the connector "$\wedge$".

4 COMPUTATIONAL COMPLEXITY

Let us now turn to the complexity issue. We obtained the following result:

Proposition 1 *Let* $\Delta^{d,f,g}$ *be a distance-based merging operator. Given a knowledge set* E *and two formulas* IC *and* α*:*

- *If* d, f *and* g *are computable in polynomial time, then determining whether* $\Delta_{IC}^{d,f,g}(E) \models \alpha$ *holds is in* Δ_2^p.
- *If* d, f *and* g *are computable in polynomial time and are polynomially bounded, then determining whether* $\Delta_{IC}^{d,f,g}(E) \models \alpha$ *holds is in* Θ_2^p.

A sketch of proof is given in the Appendix. See [Konieczny *et al.*, 2001] for a detailed proof.

As shown by the previous proposition, improving the generality of the model-based merging operators framework through an additional aggregation step does not result in a complexity shift (the decision problem for query entailment is still at the first level of PH).

We have also identified the complexity of query entailment from a merged base for the following distance-based merging operators:

Proposition 2 *Given a knowledge set* E *and two formulas from* $PROP_{PS}$ IC *and* α*, the complexity of* $\Delta_{IC}^{d,f,g}(E) \models^? \alpha$ *is reported in Tables 2, 3 and 4 (when* X *is a complexity class,* X*-c means* X*-complete).*

Table 2: Complexity results ($d = d_D$)

f/g	max	sum	lex	WS_q	OWS_q
max	Θ_2^p-c	Θ_2^p-c	Θ_2^p-c	Δ_2^p-c	Θ_2^p-c
sum	Θ_2^p-c	Θ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c
WS_q	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c
OWS_q	Θ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c

Table 3: Complexity results ($d = d_H$)

f/g	max	sum	lex	WS_q	OWS_q
max	Θ_2^p-c	Θ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c
sum	Θ_2^p-c	Θ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c
WS_q	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c
OWS_q	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c

Sketches of the proofs are given in the Appendix (again, see [Konieczny *et al.*, 2001] for fully detailed proof). It is worth adding that in the case $d = d_{H_q}$, Δ_2^p-hardness still holds whenever E is a singleton $\{K\}$, K is a singleton $\{\varphi\}$ and φ is a conjunction of variables (in this case, neither f nor g plays a significant role in the elaboration of the distance to E). As to the case $d = d_H$, Δ_2^p-hardness still holds when each explicit belief is a conjunction of variables, and Θ_2^p-hardness results hold whenever E is a singleton $\{K\}$, K is a singleton $\{\varphi\}$ and φ is a conjunction of variables.

Looking at the tables above, we can observe that the choice of the distance d has a great influence on the complexity results. Thus, whenever $d = d_H$ or $d = d_{H_q}$, the complexity results for inference from a merged base coincide whenever f (or g) is a WS_q function or a OWS_q function. This is no longer the case when $d = d_D$ is considered.

Together with Proposition 1, the complexity of many model-based merging operators already pointed out in the literature are derived as a by-product of the previous complexity results. To the best of our knowledge, the complexity of such operators has not been identified up to now[11], hence this is an additional contribution of this work. We can also note that, while the complexity of our distance-based operators is not very high (first level of PH, at most), finding out significant tractable restrictions seems a hard task since intractability is still the case in many restricted situations. Finally, our results show that some syntax-based merging operators (based on set inclusion instead of cardinality and "located" at the 2^{nd} level of PH) cannot be encoded in polynomial time as distance-based operators (unless PH collapses).

[11] However, $(\Delta_{IC}^{d_H,sum,sum}(E) \models^? \alpha) \in \Delta_2^p$ can be recovered from a complexity result given in [Liberatore and Schaerf, 2000], page 151.

Table 4: Complexity results ($d = d_{H_q}$)

f/g	max	sum	lex	WS_q	OWS_q
max	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c
sum	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c
WS_q	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c
OWS_q	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c	Δ_2^p-c

5 LOGICAL PROPERTIES

Since we aim at investigating the logical properties of our family of merging operators, a set of properties must first be considered as a base line. In [Konieczny and Pino Pérez, 1999], a study of logical properties that "good" merging operators should satisfy (in the case where all the knowledge bases are equally reliable) is carried on. The following set of postulates was proposed:

Definition 6 (IC merging operators) *Let E, E_1, E_2 be knowledge sets, K_1, K_2 be consistent knowledge bases, and IC, IC_1, IC_2 be formulas from $PROP_{PS}$. Δ is an* IC merging operator *iff it satisfies the following postulates:*

(IC0) $\Delta_{IC}(E) \models IC$

(IC1) *If IC is consistent, then $\Delta_{IC}(E)$ is consistent*

(IC2) *If $\bigwedge E$ is consistent with IC, then* $\Delta_{IC}(E) \equiv \bigwedge E \wedge IC$

(IC3) *If $E_1 \equiv E_2$ and $IC_1 \equiv IC_2$, then* $\Delta_{IC_1}(E_1) \equiv \Delta_{IC_2}(E_2)$

(IC4) *If $K_1 \models IC$ and $K_2 \models IC$, then $\Delta_{IC}(K_1 \sqcup K_2) \wedge K_1$ is consistent iff $\Delta_{IC}(K_1 \sqcup K_2) \wedge K_2$ is consistent*

(IC5) $\Delta_{IC}(E_1) \wedge \Delta_{IC}(E_2) \models \Delta_{IC}(E_1 \sqcup E_2)$

(IC6) *If $\Delta_{IC}(E_1) \wedge \Delta_{IC}(E_2)$ is consistent , then* $\Delta_{IC}(E_1 \sqcup E_2) \models \Delta_{IC}(E_1) \wedge \Delta_{IC}(E_2)$

(IC7) $\Delta_{IC_1}(E) \wedge IC_2 \models \Delta_{IC_1 \wedge IC_2}(E)$

(IC8) *If $\Delta_{IC_1}(E) \wedge IC_2$ is consistent, then* $\Delta_{IC_1 \wedge IC_2}(E) \models \Delta_{IC_1}(E)$

Two sub-classes of IC merging operators have also been defined. Majority operators that aim at resolving conflicts by listening the majority wishes, and arbitration operators that have a more consensual behaviour:

Definition 7 (majority and arbitration) *A* majority operator *is an IC merging operator that satisfies the following majority postulate:*

(Maj) $\exists n \ \Delta_{IC}(E_1 \sqcup E_2^n) \models \Delta_{IC}(E_2)$

An arbitration operator *is an IC merging operator that satisfies the following postulate:*

$$\textbf{(Arb)} \quad \left.\begin{array}{l} \Delta_{IC_1}(K_1) \equiv \Delta_{IC_2}(K_2) \\ \Delta_{IC_1 \Leftrightarrow \neg IC_2}(K_1 \sqcup K_2) \equiv (IC_1 \Leftrightarrow \neg IC_2) \\ IC_1 \not\models IC_2 \\ IC_2 \not\models IC_1 \end{array}\right\} \Rightarrow \Delta_{IC_1 \vee IC_2}(K_1 \sqcup K_2) \equiv \Delta_{IC_1}(K_1)$$

See [Konieczny and Pino Pérez, 2002; Konieczny and Pino Pérez, 1999] for more explanations about those two postulates and the behaviour of the two subclasses.

We have the following result:

Proposition 3 *$\Delta^{d,f,g}$ satisfies (IC0), (IC1), (IC2), (IC7), (IC8). The other postulates are not satisfied in the general case.*

Clearly enough, it is not the case that every distance-based merging operator is an IC merging operator (not satisfying some postulates is deliberate since we want to give some importance to the syntax in order to take into account inconsistent knowledge bases). Let us introduce some properties to be satisfied by aggregation functions f:

1) $f(x_1, \ldots, x_n) = 0$ iff $x_1 = \ldots = x_n = 0$ **(minimality)**

2) If $\varphi_1 \wedge \ldots \wedge \varphi_n$ is consistent, then $f(d(w, \varphi_1), \ldots, d(w, \varphi_n)) = f(d(w, \varphi_1 \wedge \ldots \wedge \varphi_n))$ **(and)**

3) For any permutation σ, $f(x_1, \ldots, x_n) = f(\sigma(x_1, \ldots, x_n))$ **(symmetry)**

4) If $f(x_1, \ldots, x_n) \leq f(y_1, \ldots, y_n)$, then $f(x_1, \ldots, x_n, z) \leq f(y_1, \ldots, y_n, z)$ **(composition)**

5) If $f(x_1, \ldots, x_n, z) \leq f(y_1, \ldots, y_n, z)$, then $f(x_1, \ldots, x_n) \leq f(y_1, \ldots, y_n)$ **(decomposition)**

Now, if one wants to recover the full set of postulates (IC0)-(IC8):

Proposition 4 *A distance-based merging operator $\Delta^{d,f,g}$ satisfies (IC0)-(IC8) if and only if the function f satisfies (minimality) and (and), and the function g satisfies (minimality), (symmetry), (composition) and (decomposition).*

Concerning the operators examined in the previous section, we have identified the following properties:

Proposition 5 *$\Delta^{d,f,g}$ satisfies the logical properties stated in Tables 5 and 6. Since all these operators are already known to satisfy (IC0), (IC1), (IC2), (IC7) and (IC8) (cf. Proposition 3), we avoid repeating such postulates here. For more readability, postulate (ICi) is noted i and M (resp. A) stands for (Maj) (resp. (Arb)).*

Table 5: Logical properties ($d = d_D$)

f/g	max	sum	lex	WS_q
max	3,4,5,A	3,4,5,6,M,A		5,6,M
sum	5,A	5,6,M	5,6,A	5,6,M
$WS_q - OWS_q$	5,A	5,6,M	5,6,A	5,6,M

Table 6: Logical properties ($d = d_H$ or $d = d_{H_q}$)

f/g	max	sum	lex	WS_q
max	5,A	5,6,M	5,6,A	5,6,M
sum	5,A	5,6,M	5,6,A	5,6,M
$WS_q - OWS_q$	5,A	5,6,M	5,6,A	5,6,M

The tables above show our operators to exhibit different properties. We remark that among our operators, only $\Delta^{d_D,max,sum}$ satisfies all listed properties. Failing to satisfy (IC3) (*irrelevance to the syntax*) in many cases is not surprising, since we want to allow our operators to take syntax into account. (IC4) imposes that, when merging two knowledge bases, if the result is consistent with one knowledge base, it has to be consistent with the other one – this fairness postulate is irrelevant when working with non-symmetric operators (so, unsurprisingly, it is not satisfied for $g = WS_q$). This postulate is not satisfied by any operator for which d is Hamming distance since cardinalities of the knowledge bases have an influence on f, and more generally, it is hardly satisfiable when working with syntax-dependent operators. (IC5) and (IC6) are related to Pareto dominance in social choice theory and are really important for multi-source aggregation; so it is worth noting that almost all operators satisfy them (only operators for which $g = max$ do not satisfy (IC6)).

We do not put the operators with $g = OWS_q$ in the tables because they gather many aggregation functions and so they do not satisfy a lot of logical properties. Moreover, some properties (as (IC5) and (IC6)) require to be able to cope with knowledge sets of different sizes, whereas $g = OWS_q$ operators have to specify

exactly the size of the knowledge sets. It is possible to generalize the definition of those operators to cope with these cases but it is out of the scope of this paper.

6 CONCLUSION

The major contribution of this paper is a new framework for propositional merging. It is general enough to encompass many existing operators (both model-based ones and syntax-based ones) and to enable the definition of many new operators (symmetric or not). Both the logical properties and the computational properties of the merging operators pertaining to our framework have been investigated. Some of our results are large-scope ones in the sense that they make sense under very weak conditions on the three parameters that must be set to define an operator in our framework. By instantiating our framework and considering several distances and aggregation functions, more refined results have also been obtained.

This work calls for several perspectives. One of them consists in analyzing the properties of the distance-based operators that are achieved when some other aggregation functions or some other distances are considered. For instance, suppose that a collection of formulas of interest (topics) is available. In this situation, the distance between ω_1 and ω_2 can be defined as the number of relevant formulas on which ω_1 and ω_2 differs (i.e., such that one of them satisfies the formula and the other one violates it). Several additional distances could also be defined and investigated (see e.g. [Lafage and Lang, 2001] for distances based on Choquet integral).

Acknowledgements

The third author has been partly supported by the IUT de Lens, the Université d'Artois, the Région Nord/Pas-de-Calais under the TACT-TIC project, and by the European Community FEDER Program.

Appendix: Proof sketches of the complexity results

Sketch of Proof of Proposition 1 : These results are consequences of the two following lemmata:

Lemma 1 *Let k be an integer; if d, f and g are computable in polynomial time, then the problem of determining whether* $\min_{\omega \models IC} d(\omega, E) \leq k$ *given IC, E and k is in* NP.

Proof : It is sufficient to consider the following non-deterministic algorithm:

1. guess an interpretation ω and N interpretations $\omega_{i,j}$ ($i = 1..m$, $j = 1..n_i$) over $Var(E \cup \{IC\})$, where $N = \Sigma_{i=1..n} n_i$ is the total number of formulas $\varphi_{i,j}$ in E;
2. check that $\omega \models IC$ and that $\omega_{i,j} \models \varphi_{i,j}$ for all $i = 1..m$ and all $j = 1..n_i$;
3. compute $d(\omega, \omega_{i,j})$ for all i and all j;
4. compute $d(\omega, K_i)$ for all i;
5. compute $d(\omega, E)$ and check that $d(\omega, E) \leq k$.

This algorithm runs in polynomial time in the size of the input (E, IC and k represented in binary) since d, f, g are computable in polynomial time. □

Lemma 2 *If for all $\omega \in \mathcal{W}$ the value of $d(\omega, E)$ is bounded by the value $h(|E| + |IC|)$ (where h is a function with values in $\mathbb{N}$) then the value* $\min_{\omega \models IC} d(\omega, E)$ *can be computed using $\lceil \log h(|E| + |IC|) \rceil$ calls to an* NP *oracle.*

Proof : $min = \min_{\omega \models IC} d(\omega, E)$ can be computed using binary search on $\{0, \ldots, h(|E| + |IC|)\}$ with at each step a call to an NP oracle to check whether $\min_{\omega \models IC} d(\omega, E) \leq k$ (that is in NP from lemma 1). Since a binary search on $\{0, \ldots, h(|E| + |IC|)\}$ needs at most $\lceil \log h(|E| + |IC|) \rceil$ steps, the result follows. □

• *Point 1. of Proposition 1*
If d, f and g are computable in polynomial time, then for every knowledge set E and every $\omega \in \mathcal{W}$, the binary representation of $d(\omega, E)$ is bounded by $p(|E| + |IC|)$, where p is a polynomial. Hence, the value of $d(\omega, E)$ is bounded by $2^{p(|E|+|IC|)}$. From lemma 2, we can conclude that min can be computed using a polynomial number of calls to an NP oracle. Now, let E be a knowledge set, IC be a formula, k be an integer and α be a formula, it can be shown that the problem of determining whether there exists a model ω of IC such that $d(\omega, E) = k$ and such that $\omega \not\models \alpha$ is in NP. So we can show that $\Delta_{IC}^{d,f,g}(E) \not\models \alpha$ using first a polynomial number of calls to an NP oracle in order to compute min, and then using an additional call to an NP oracle in order to determine whether there exists a model ω of IC s.t. $d(\omega, E) = min$ and $\omega \not\models \alpha$. Hence the membership to

Δ_2^p for this problem, and hence for its complement.

• *Point 2. of Proposition 1*
When d, f and g are polynomially bounded, the proof is similar to the one of point 1., but the computation of $\min_{\omega \models IC} d(\omega, E)$ needs only a logarithmic number of steps since h is polynomially bounded, hence the membership to Θ_2^p.

□

Sketch of Proof of Proposition 2 :

1. Membership
Membership-to-Δ_2^p results are direct consequences of Proposition 1 since both distances and aggregation functions can be computed in polynomial time.
Membership-to-Θ_2^p results are also consequences of Proposition 1, except those for which f or g is an OWS_q (including lex) when the drastic distance d_D is considered; these cases are briefly discussed now:

- *case* $d = d_D$, $f = max$ *and* $g = OWS_q$. We first establish that $d(\omega, E)$ can only take only a polynomial number of different values, and that this set of possible values can be computed in polynomial time. Indeed, if $k_E(\omega)$ is the number of belief bases K_i from E s.t. $\omega \models K_i$, we have $d(\omega, E) = \sum_{i=k(\omega)+1}^{m} q_i$; which makes $|E| + 1$ different values, computable in polynimial time. The rest of the proof is similar to the proof of membership to Θ_2^p in the cases where g and g are polynomially bounded, the difference being here that the minimal value $min = \min_{\omega \in \Omega} d(\omega, E)$ is computed through binary search using the precomputed $|E|+1$ different possible values for $d(\omega, E)$.

- the *case* $d = d_D$, $f = OWS_q$ *and* $g = max$ is similar, the main difference is that $d(\omega, E)$ can only take at most $max_{i \in 1 \ldots n} card(K_i)$ different values.

Finally, as to the basic merging operator (d_D, max, max), determining whether a formula α is a logical consequence of the merged base E given IC can be achieved using the following algorithm:
`if` $sat(E \cup \{IC\})$ `then return`$(unsat(E \cup \{IC, \neg\alpha\}))$ `else return`$(unsat(\{IC, \neg\alpha\}))$, which shows membership of the decision problem to BH_2.

2. Hardness:

- *table 2,* Θ_2^p*-hardness results*: they are direct consequences of hardness results for cardinality-maximizing base revision $\circ_C$ (Theorem 5.14 from [Nebel, 1998]) since we have $\Delta_{IC}^{d_D,f,g}(\{\{\varphi_1\}, \ldots, \{\varphi_n\}\}) \equiv \{\varphi_1, \ldots, \varphi_n\} \circ_C IC$ for any $\langle f, g \rangle \in \{\langle max, max \rangle, \langle max, lex \rangle, \langle sum, max \rangle, \langle sum, sum \rangle\}$. Since sum is a specific OWS_q function, the corresponding results still hold in the cases $(f = OWS_q, g = max)$ and $(f = max, g = OWS_q)$.

- *table 2, case* $d = d_D$, $f = OWS_q$, $g = sum$: Δ_2^p-hardness is established by considering the following polynomial reduction from the Δ_2^p-complete problem MAX-SAT-ASG$_{odd}$ [Wagner, 1987]. MAX-SAT-ASG$_{odd}$ is the following decision problem: given a propositional formula Σ s.t. $Var(\Sigma) = \{x_1, \ldots, x_n\}$ and a strict ordering $x_1 < x_2 < \ldots < x_n$ on $Var(\Sigma)$ inducing the lexicographic ordering $\preccurlyeq$ on Ω, is the greatest model ω of Σ w.r.t. $\preccurlyeq$ such that $\omega(x_n) = 1$? We just give here the reduction: to Σ s.t. $Var(\Sigma) = \{x_1, \ldots, x_n\}$, we associate the tuple $M(\Sigma) = \langle E, IC, \alpha \rangle$, where $E = \{K_i \mid i \in 1 \ldots n\}$, $IC = \Sigma$, $\alpha = x_n$ and for each $i \in 1 \ldots n$, $K_i = \{\bigwedge_{k=1}^{n+2-j} x_i \mid j \in 1 \ldots n+2-i\}$ (each K_i contains $n+2-i$ formulas that are syntactically distinct but all equivalent to x_i), and we consider the OWS_q function f induced by q s.t. $q(1) = 1$ and for every $j > 1$, $q(j) = 2^{i-2}$.

- *table 2,* Δ_2^p*-hardness results in the case* $f = sum$: hardness in the case $(d = d_D, f = sum, g = lex)$ is easily derived by taking advantage of the Δ_2^p-hardness result in the case where each K_i is a singleton reduced to a conjunction of atoms (hence f is irrelevant), $g = lex$ and $d = d_H$. Since sum is a specific WS_q function and lex is a specific OWS_q function, this hardness result can be extended to the rest of the table, except for the cases where f is a WS_q function and $g \in \{max, sum\}$) and where g is a WS_q function. In the latter case, the Δ_2^p-hardness of linear base revision $\circ_L$ (Theorem 5.9 from [Nebel, 1998]) can be used to obtain the desired result: indeed, it is sufficient to consider belief bases K_i reduced to singletons; we have $\Delta_{IC}^{d_D,f,g}(\{K_1, \ldots, K_n\}) \equiv \{K_1, \ldots, K_n\} \circ_L IC$, where g is the weighted sum induced by q s.t. $q(i) = 2^{n-i}$, and each K_i is viewed as the unique formula it contains. Here, the preference ordering over $\{K_1, \ldots, K_n\}$ is s.t. $K_1 < K_2 < \ldots < K_n$.

- *table 2, case* $(d = d_D, f = g = max)$: it is sufficient to consider the following polynomial reduction M from SAT-UNSAT: to a pair of formulas $\langle \varphi, \psi \rangle$ which do not share variables (this can be assumed without loss of generality), we let $M(\langle \varphi, \psi \rangle) = \langle E = \varphi, IC = new, \alpha = \varphi \wedge new \wedge$

$\neg\psi\rangle$ where *new* is a new variable and we check that $\langle\varphi,\psi\rangle \in$ SAT-UNSAT iff α is a logical consequence of the merged base E given IC.

- *table 3, Θ_2^p-hardness results.* They still hold in the situation where E contains only one belief base K and K itself contains only one formula that is a conjunction of atoms. This merely shows that our hardness result is independent from f and g (since they are irrelevant whenever E and K are singletons) but is a consequence of the distance that is used (Hamming). Indeed, in this restricted case, $\Delta_{IC}^{d_H,f,g}(\{K\})$ is equivalent to $K \circ_D IC$ where $\circ_D$ is Dalal's revision operator. The fact that the inference problem from $K \circ_D IC$ is Θ_2^p-hard (even in the restricted case where K is a conjunction of atoms) concludes the proof (see Theorem 6.9 from [Eiter and Gottlob, 1992]).
- *table 3, Δ_2^p-hardness results.* We show that these Δ_2^p-hardness results hold in the restricted case where each K_i is a singleton, reduced to a conjunction of literals (hence f is irrelevant) when $g = lex$ by the following polynomial reduction M from MAX-SAT-ASG$_{odd}$: to any propositional formula Σ s.t. $Var(\Sigma) = \{x_1, \ldots, x_n\}$ we associate $M(\Sigma) = \langle E = \{K_i = \{x_i \wedge \bigwedge_{j=i+1}^{2n-i+1} new_j\} \mid i \in 1 \,..\, n\}, IC = \Sigma \wedge \bigwedge_{j=2}^{2n} \neg new_j, \alpha = x_n\rangle$ where each new_j $(j \in 2 \,..\, 2n)$ is a new variable.
- *table 4.* We show that Δ_2^p-hardness holds in the very restricted case where E contains only one belief base K and K itself contains only one formula that is a conjunction of atoms. This merely shows that our hardness result is independent from f and g (since they are irrelevant whenever E and K are singletons) but is a consequence of the family of distances that is used (weighted Hamming). This is done by the following polynomial reduction M from MAX-SAT-ASG$_{odd}$: to any Σ s.t. $Var(\Sigma) = \{x_1, \ldots, x_n\}$ we associate $M(\Sigma) = \langle E = \{\{\bigwedge_{i=1}^{n} x_i\}\}, IC = \Sigma, \alpha = x_n\rangle$ and the weighted Hamming distance d_{H_q} induced by q s.t. $\forall i \in 1 \,..\, n$, $q(x_i) = 2^{n-i}$.

□

References

[Baral *et al.*, 1991] C. Baral, S. Kraus, and J. Minker. Combining multiple knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 3(2):208–220, 1991.

[Baral *et al.*, 1992] 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.

[Benferhat *et al.*, 2000] S. Benferhat, D. Dubois, S. Kaci, and H. Prade. Encoding information fusion in possibilistic logic: a general framework for rational syntactic merging. In *Proc. of ECAI'00*, pages 3–7, 2000.

[Bloch and Hunter, 2001] I. Bloch and A. Hunter, editors. *Fusion: General Concepts and Characteristics*, volume 16 of *International Journal of Intelligent Systems*. Wiley, 2001. Special Issue on Data and Knowledge Fusion.

[Dalal, 1988] M. Dalal. Investigations into a theory of knowledge base revision: preliminary report. In *Proc. of AAAI'88*, pages 475–479, 1988.

[Eiter and Gottlob, 1992] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision , updates, and counterfactuals. *Artificial Intelligence*, 57(2-3):227–270, 1992.

[Hansson, 1998] S. O. Hansson. Revision of belief sets and belief bases. *Handbook of Defeasible Reasoning and Uncertainty Management Systems, Vol. 3*, pages 17–75, 1998.

[Konieczny and Pino Pérez, 1999] S. Konieczny and R. Pino Pérez. Merging with integrity constraints. In *Proc. of ECSQARU'99*, LNAI 1638, pages 233–244, 1999.

[Konieczny and Pino Pérez, 2002] S. Konieczny and R. Pino Pérez. On the frontier between arbitration and majority. In *Proc. of KR'02*, 2002.

[Konieczny *et al.*, 2001] S. Konieczny, J. Lang, and P. Marquis. Distance-based merging: a general framework and some complexity results. Technical report, IRIT, `ftp://ftp.irit.fr/pub/IRIT/RPDMP/DBMC.ps.gz`, 2001.

[Konieczny, 2000] S. Konieczny. On the difference between merging knowledge bases and combining them. In *Proc. of KR'00*, pages 135–144, 2000.

[Lafage and Lang, 2000] C. Lafage and J. Lang. Logical representation of preferences for group decision theory. In *Proc. of KR'00*, pages 457–468, 2000.

[Lafage and Lang, 2001] C. Lafage and J. Lang. Propositional distances and preference representation. In *Proc. of ECSQARU'01*, pages 48–59, 2001.

[Liberatore and Schaerf, 1998] P. Liberatore and M. Schaerf. Arbitration (or how to merge knowledge bases). *IEEE Transactions on Knowledge and Data Engineering*, 10(1):76–90, 1998.

[Liberatore and Schaerf, 2000] P. Liberatore and M. Schaerf. Brels: a system for the integration of knowledge bases. In *Proc. of KR'00*, pages 145–152, 2000.

[Lin and Mendelzon, 1999] J. Lin and A. O. Mendelzon. Knowledge base merging by majority. In *Dynamic Worlds: From the Frame Problem to Knowledge Management*. Kluwer, 1999.

[Nebel, 1989] B. Nebel. A knowledge level analysis of belief revision. In *Proc. of KR'89*, pages 301–311, 1989.

[Nebel, 1998] B. Nebel. How hard is it to revise a belief base? *Handbook of Defeasible Reasoning and Uncertainty Management Systems, Vol. 3: Belief Change*, pages 77–145, 1998.

[Papadimitriou, 1994] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Pinkas, 1995] G. Pinkas. Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge. *Artificial Intelligence*, 77:203–247, 1995.

[Revesz, 1997] P. Z. Revesz. On the semantics of arbitration. *International Journal of Algebra and Computation*, 7(2):133–160, 1997.

[Wagner, 1987] K. W. Wagner. More complicated questions about maxima and minima, and some closures of NP. *Theoretical Computer Science*, 51:53–80, 1987.

[Yager, 1998] R. R. Yager. On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18:183–190, 1998.

On the Frontier between Arbitration and Majority

Sébastien Konieczny
IRIT
Université Paul Sabatier
31062 Toulouse Cedex - France
konieczny@irit.fr

Ramón Pino Pérez
Departamento de Matemáticas
Facultad de Ciencias
Universidad de los Andes
Mérida - Venezuela
pino@ciens.ula.ve

Abstract

We give in this paper new results on merging operators. Those operators aim to define the beliefs (or goals) of an agents' group from the individuals beliefs (goals). Using the logical framework of [KP99] we study the relationships between two important sub-families of merging operators: majority operators and arbitration operators. An open question was to know if those two families are disjoint or not. We show that there are operators that belong simultaneously to the two families. Furthermore the new family introduced allows the user to choose the "consensual level" he wants for his majority operator.

1 Introduction

When several agents interact in order to achieve a common task, they have to agree from time to time on what are the beliefs (or the goals) of the group. When some agents disagree on these common beliefs (goals), then one has to enter in a *negotiation* process. The problem is that sometimes the negotiation step do not rule out all the conflicts. But, even in this case, the group has to take a decision on what are its beliefs (goals) to carry on. So, in such cases, an *aggregation* step is needed between agents wishes.

So, formally, when a decision has to be taken about beliefs (goals) of the group, we can consider this as a two step process. First, a *negotiation* step allows agents to try to convince undecided or opponents. Then, when all agents have fixed opinions, an *aggregation* step states what are the common beliefs (goals) of the group.

The first step of this process has been extensively studied in multi-agents works (see *e.g.* [APM00] for an example of formalisation of negotiation by means of argumentation). But the second one is usually only quickly quoted. Indeed, in most of those works, when some conflict is not solved after the negotiation step, one uses expeditious means to solve the conflicts. For example, by supposing the existence of some oracle that decides what is the good solution, or by using a preference relation between agents denoting the relative reliability of each source. But, even if those solutions often allow to rule out the conflicts, the basic problem is not solved and there still are problems in some cases. For example, it is not realistic to suppose that an oracle exists and always knows the good answer. And in the case of using a reliability ordering, there are cases where some equally reliable agents disagree and we are back with our basic problem.

The formal framework for solve this belief (goals) *aggregation* step, is the use of knowledge merging operators [CH97, Cho98, BKMS92, LM99, Rev97, Kon00].

In some related works different sets of logical properties that knowledge merging operators have to satisfy have been proposed [Rev97, LS98, LM99, KP98, KP99]. Those logical characterisations are used to define a taxonomy of merging operators, that allows to compare different merging methods and to choose the method corresponding to the behaviour wanted in a particular application.

We will focus on the *merging with integrity constraints* characterisation given in [KP99, KP02]. This characterisation allows to make a distinction between two major sub-classes of merging operators: majority operators and arbitration operators. Majority operators solve conflicts using majority wishes, that is they try to satisfy the group as a whole. Whereas arbitration operators have a more consensual behaviour, trying to satisfy each agent as far as possible.

Consider the following example to illustrate these two behaviours (the following example is stated in terms of

goals, but one can find similar ones concerning beliefs):

Example 1 *Ally, Brian and Charles have to decide what they will do this night. Brian and Ally want to go to the restaurant and to the cinema. Charles does not want to go out this night and so he does not want to go nor to the restaurant nor to the cinema. Taking some majority merging operator the result of the merging is that the group will decide to go both to the restaurant and to the cinema, whereas Charles would certainly have a bad night. If one takes an arbitration operator the result will be that the group has to decide to go either to the cinema or to the cinema but not both. So each member of the group will be satisfied as much as possible.*

So these two sub-classes have very different conflict resolution policies. An open question was to know if these two sub-classes are disjoint or not. And, though it seems natural to bet on a strict partition, we show in this paper that it is not the case. That is, there exists operators that belong simultaneously to the two sub-classes. We first give a trivial operator that straightforwardly satisfy this condition. But the real question was to know if more complex operators can satisfy it too. We show that, in the finite case, a whole family of (non-trivial) operators are both arbitration and majority operators. The new family of operators introduced, generalisation of a well known majority merging method [Rev97, LM99, KP99], allow to choose the "consensual level" that best fit the application needs.

The paper is organised as follow. In section 2, we give the definition of merging with integrity constraints operators, arbitration and majority operators are also defined. Then, we give in section 3 some concrete operators in order to illustrate the differences of behaviour between arbitration and majority operators. In section 4, we show that it is possible for an operator to be both a majority and an arbitration operator. We discuss briefly in section 5 of alternative expressions of arbitration behaviour. We conclude in section 6 with some open questions.

2 Merging with Integrity Constraints

We consider a propositional language $\mathcal{L}$ over a finite alphabet $\mathcal{P}$ of propositional atoms. An interpretation is a function from $\mathcal{P}$ to $\{0,1\}$. The set of all the interpretations is denoted $\mathcal{W}$. An interpretation I is a model of a formula if and only if it makes it true in the usual classical truth functional way. Let φ be a formula, $mod(\varphi)$ denotes the set of models of φ, *i.e.* $mod(\varphi) = \{I \in \mathcal{W} \; I \models \varphi\}$.

A *knowledge base* φ is a finite set of propositional formulae.

Let $\varphi_1, \ldots, \varphi_n$ be n knowledge bases (not necessarily different). We call *knowledge set* the multi-set Ψ consisting of those n knowledge bases: $\Psi = \{\varphi_1, \ldots, \varphi_n\}$. We note $\bigwedge \Psi$ the conjunction of the knowledge bases of Ψ, i.e. $\bigwedge \Psi = \varphi_1 \wedge \cdots \wedge \varphi_n$. The union of multi-sets will be noted $\sqcup$.

By abuse if φ is a knowledge base, φ will also denote the knowledge set $\Psi = \{\varphi\}$. For a positive integer n we will denote Ψ^n the multi-set when Ψ appears n times.

Definition 1 *Let Ψ_1, Ψ_2 be two knowledge sets. Ψ_1 and Ψ_2 are equivalent, noted $\Psi_1 \leftrightarrow \Psi_2$, iff there exists a bijection f from $\Psi_1 = \{\varphi_1^1, \ldots, \varphi_n^1\}$ to $\Psi_2 = \{\varphi_1^2, \ldots, \varphi_n^2\}$ such that $\vdash f(\varphi) \leftrightarrow \varphi$.*

A pre-order $\leq$ is a reflexive and transitive relation. A pre-order is total if $\forall I, J \; I \leq J$ or $J \leq I$. Let $\leq$ be a pre-order, we define $<$ as follows: $I < J$ iff $I \leq J$ and $J \not\leq I$, and $\simeq$ as $I \simeq J$ iff $I \leq J$ and $J \leq I$. We wrote $I \in min(mod(\varphi), \leq)$ iff $I \models \varphi$ and $\forall J \in mod(\varphi) \; I \leq J$.

Once these definitions are stated, we can define merging operators. A knowledge base φ will denote the beliefs [1] of an agent. A knowledge set Ψ will denote a group of agents. The aim of merging operators is to define what are the beliefs of the group from the individuals beliefs and the constraints imposed by the system (physical constraints, laws, etc.). So, a merging operator Δ is a function that maps a knowledge set Ψ and a knowledge base μ that denotes the integrity constraints of the system, to a knowledge base $\Delta_\mu(\Psi)$ that contains the beliefs of the group. Recall that we suppose that all the knowledge bases have the same importance (*i.e.* reliability, hierarchical importance, etc...) and that they denote the beliefs (goals) of independent sources (agents). See *e.g.* [BDL+98, Cho98] for examples on prioritised knowledge bases.

The logical properties that one could expect from a belief merging operator are [KP99]:

Definition 2 *Δ is a merging with integrity constraints operator (IC merging operator in short) if and only if it satisfies the following properties:*

(IC0) $\Delta_\mu(\Psi) \models \mu$

(IC1) *If μ is consistent, then $\Delta_\mu(\Psi)$ is consistent*

(IC2) *If $\bigwedge \Psi$ is consistent with μ, then* $\Delta_\mu(\Psi) \equiv \bigwedge \Psi \wedge \mu$

[1] in the following, we will call "beliefs" the beliefs or the goals of an agent

(IC3) *If* $\Psi_1 \equiv \Psi_2$ *and* $\mu_1 \equiv \mu_2$, *then* $\Delta_{\mu_1}(\Psi_1) \equiv \Delta_{\mu_2}(\Psi_2)$

(IC4) *If* $\varphi_1 \models \mu$ *and* $\varphi_2 \models \mu$, *then* $\Delta_\mu(\varphi_1 \sqcup \varphi_2) \wedge \varphi_1$ *is consistent iff* $\Delta_\mu(\varphi_1 \sqcup \varphi_2) \wedge \varphi_2$ *is consistent*

(IC5) $\Delta_\mu(\Psi_1) \wedge \Delta_\mu(\Psi_2) \models \Delta_\mu(\Psi_1 \sqcup \Psi_2)$

(IC6) *If* $\Delta_\mu(\Psi_1) \wedge \Delta_\mu(\Psi_2)$ *is consistent , then* $\Delta_\mu(\Psi_1 \sqcup \Psi_2) \models \Delta_\mu(\Psi_1) \wedge \Delta_\mu(\Psi_2)$

(IC7) $\Delta_{\mu_1}(\Psi) \wedge \mu_2 \models \Delta_{\mu_1 \wedge \mu_2}(\Psi)$

(IC8) *If* $\Delta_{\mu_1}(\Psi) \wedge \mu_2$ *is consistent, then* $\Delta_{\mu_1 \wedge \mu_2}(\Psi) \models \Delta_{\mu_1}(\Psi)$

The intuitive meaning of the properties is the following: (IC0) assures that the result of the merging satisfies the integrity constraints. (IC1) states that if the integrity constraints are consistent, then the result of the merging will be consistent. (IC2) states that if possible, the result of the merging is simply the conjunction of the knowledge bases with the integrity constraints. (IC3) is the principle of irrelevance of syntax, expressing the fact that the result of the merging has to depend only of the expressed opinions and not of their syntactical presentation. (IC4) is the fairness postulate, the point is that when we merge two knowledge bases, merging operators must not give preference to one of them. (IC5) expresses the following idea: if a group Ψ_1 compromises on a set of alternatives which A belongs to, and another group Ψ_2 compromises on another set of alternatives which contains A too, so A has to be in the chosen alternatives if we join the two groups. (IC5) and (IC6) together state that if you could find two subgroups which agree on at least one alternative, then the result of the global merging will be exactly those alternatives the two groups agree on. (IC7) and (IC8) state that the notion of closeness is well-behaved, i.e. that an alternative that is preferred among the possible alternatives (μ_1), will remain preferred if we restrict the possible choices ($\mu_1 \wedge \mu_2$).

One can notice that when the knowledge set is a singleton (i.e. $\Psi = \{\varphi\}$) doing the merging $\Delta_\mu(\{K\})$ is exactly the revision of φ by a new evidence μ. That is $\Delta_\mu(\{K\}) = \varphi \circ \mu$, where $\circ$ is an AGM belief revision operator [Gär88, AGM85, KM91]. Thus IC merging operators can be considered as a generalisation of belief revision operators. See [KP02] for more results on the relationship between merging and belief revision.

We will now define the two major sub-classes of merging operators: majority and arbitration operators : An IC merging operator is a majority operator if it satisfies the following property:

(Maj) $\exists n \ \Delta_\mu(\Psi_1 \sqcup \Psi_2^n) \vdash \Delta_\mu(\Psi_2)$

This postulate expresses the fact that if an opinion has a large audience, it will be the opinion of the group. So, majority operators try to satisfy the group as a whole[2]. On the other hand, arbitration operators try to satisfy each agent as far as possible : An IC merging operator is an arbitration operator if it satisfies the following property:

$$\textbf{(Arb)} \quad \left.\begin{array}{l} \Delta_{\mu_1}(\varphi_1) \leftrightarrow \Delta_{\mu_2}(\varphi_2) \\ \Delta_{\mu_1 \leftrightarrow \neg\mu_2}(\varphi_1 \sqcup \varphi_2) \leftrightarrow (\mu_1 \leftrightarrow \neg\mu_2) \\ \mu_1 \not\vdash \mu_2 \\ \mu_2 \not\vdash \mu_1 \end{array}\right\} \Rightarrow$$
$$\Delta_{\mu_1 \vee \mu_2}(\varphi_1 \sqcup \varphi_2) \leftrightarrow \Delta_{\mu_1}(\varphi_1)$$

This postulate ensures that this is the median possible choices that are preferred. It is much more intuitive when it is expressed in terms of syncretic assignment (*cf* condition 8 below). The point is that we can improve the result for one of the member of the group only if it does not make the result worth for an other member. This kind of behaviour is very close to egalitarism in social choice theory (see *e.g.* [Mou88]).

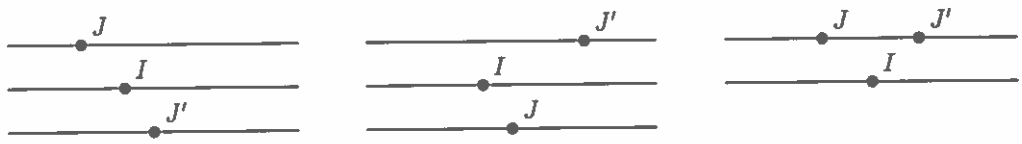
We will illustrate the (Arb) requirements on the following scenario:

Example 2 *Tom and David missed the soccer match yesterday between reds and yellows. So they don't know the result of the match. Tom listened in the morning that reds made a very good match. So he thinks that a win of reds is more plausible than a draw and that a draw is more reliable than a win of yellows. David was told that after that match yellows have now a lot of chances of winning the championship. From this information he infers that yellows win the match, or otherwise at least take a draw. Confronting their point of view, Tom and David agree on the fact that the two teams are of the same strength, and that they had the same chances of winning the match. What arbitration demand is that, with those informations, Tom and David have to agree that a draw between the two teams is the more plausible result.*

Now we will give a representation theorem for those operators in terms of pre-orders on interpretations. It provides a more constructive definition of those operators. We need first some definitions:

Definition 3 *A* syncretic assignment *is a function mapping each knowledge set* Ψ *to a total pre-order* $\leq_\Psi$

[2]Remark that, with the other postulates, (Maj) implies some kind of monotony property when the number of proponents grows: $\exists n_0 \ \forall n > n_0 \ \Delta_\mu(\Psi_1 \sqcup \Psi_2^n) \vdash \Delta_\mu(\Psi_2)$.



(a) φ_1 (b) φ_2 (c) $\varphi_1 \sqcup \varphi_2$

Figure 1: Arbitration

over interpretations such that for any knowledge sets Ψ, Ψ_1, Ψ_2 *and for any knowledge bases* φ_1, φ_2*:*

1. *If* $I \models \Psi$ *and* $J \models \Psi$*, then* $I \simeq_\Psi J$
2. *If* $I \models \Psi$ *and* $J \not\models \Psi$*, then* $I <_\Psi J$
3. *If* $\Psi_1 \equiv \Psi_2$*, then* $\leq_{\Psi_1} = \leq_{\Psi_2}$
4. $\forall I \models \varphi_1 \ \exists J \models \varphi_2 \ J \leq_{\varphi_1 \sqcup \varphi_2} I$
5. *If* $I \leq_{\Psi_1} J$ *and* $I \leq_{\Psi_2} J$*, then* $I \leq_{\Psi_1 \sqcup \Psi_2} J$
6. *If* $I <_{\Psi_1} J$ *and* $I \leq_{\Psi_2} J$*, then* $I <_{\Psi_1 \sqcup \Psi_2} J$

A majority syncretic assignment *is a syncretic assignment which satisfies the following:*

7. *If* $I <_{\Psi_2} J$*, then* $\exists n \ I <_{\Psi_1 \sqcup \Psi_2{}^n} J$

A fair syncretic assignment *is a syncretic assignment which satisfies the following:*

8. $\left.\begin{array}{l} I <_{\varphi_1} J \\ I <_{\varphi_2} J' \\ J \simeq_{\varphi_1 \sqcup \varphi_2} J' \end{array}\right\} \Rightarrow I <_{\varphi_1 \sqcup \varphi_2} J$

The two first conditions ensure that the models of the knowledge set (if any) are the more plausible interpretations for the pre-order associated to the knowledge set. The third condition states that two equivalent knowledge sets have the same associated pre-orders. Those three conditions are very closed to the ones existing in belief revision for faithful assignments [KM91]. The fourth condition states that, when merging two belief bases, for each model of the first one, there is a model of the second one that is at least as good than the first one. It ensures that the two knowledge bases are given the same consideration. The fifth condition says that if an interpretation I is at least as plausible as an interpretation J for a knowledge set Ψ_1 and if I is at least as plausible as J for a knowledge set Ψ_2, then if one joins the two knowledge sets, then I will still be at least as plausible as J. The sixth condition strengthen the previous condition by saying that an interpretation I is at least as plausible as an interpretation J for a knowledge set Ψ_1 and if I is strictly more plausible than J for a knowledge set Ψ_2, then if one joins the two knowledge sets, then I will be strictly more plausible than J. These two previous conditions corresponds to Pareto conditions in Social Choice Theory [Arr63, Kel78]. Condition 7 says that if an interpretation I is strictly more plausible than an interpretation J for a knowledge set Ψ_2, then there is a quorum n of repetitions of the knowledge set from which I will be more plausible than J for the larger knowledge set $\Psi_1 \sqcup \Psi_2{}^n$. This condition seems to be the weakest form of "majority" condition one could state. Condition 8 states that if an interpretation I is more plausible than an interpretation J for a belief base φ_1, if I is more plausible than J' for an other base φ_2, and if J and J' are equally plausible for the knowledge set $\varphi_1 \sqcup \varphi_2$, then I has to be more plausible than J and J' for $\varphi_1 \sqcup \varphi_2$. This requirement is illustrated figure 1 (the lower appears an interpretation, the more preferred it is). See also Example 2 for an intuitive explanation.

And the representation theorem is:

Theorem 1 *An operator is an IC merging operator (respectively IC majority merging operator or IC arbitration operator) if and only if there exists a syncretic assignment (respectively majority syncretic assignment or fair syncretic assignment) that maps each knowledge set* Ψ *to a total pre-order* $\leq_\Psi$ *such that*

$$mod(\triangle_\mu(\Psi)) = \min(mod(\mu), \leq_\Psi)$$

This theorem shows that a merging operator corresponds to a family of pre-orders. In fact, a lot of operators are defined directly from those pre-orders, using a function that maps each knowledge set to a pre-order. It is the case with all operators defined from a distance. We give some of them in the following section.

3 Some IC merging operators

We give in this section the definition of three families of operators. All those operators are based on a distance between interpretations that induces the pre-order associated to each knowledge set. We define also a new family of operators, that generalises the $\Delta^{d,\Sigma}$ family.

Let d be a distance between interpretations[3], that is a function $d : \mathcal{W} \times \mathcal{W} \mapsto I\!N$ such that :

- $d(I, J) = d(J, I)$
- $d(I, J) = 0$ iff $I = J$

For example, one can use the Dalal distance [Dal88], noted, d_H, that is the Hamming distance between two interpretations (the number of propositional letters on which the two interpretations differ). We will use this distance in the examples because it is a well known, easy to define, distance but one has to keep in mind that it is not the sole possible choice and that the logical properties do not depend of the chosen distance.

This distance between interpretations induces naturally a distance between an interpretation and a knowledge base as follows:

$$d(I, \varphi) = \min_{J \models \varphi} d(I, J)$$

The difference between the four families of operators we define next lie in the way this distance between an interpretation and a knowledge base is used in order to define the distance between an interpretation and the knowledge set. So, it is this aggregation step of the individual preferences (distances) in a global one that makes behaviour differences between the families.

The three families stated next are well known, the $\Delta^{d,Max}$ family has been used in [Rev93, Rev97], the $\Delta^{d,\Sigma}$ family in [Rev97, LM99, KP99], and the $\Delta^{d,GMax}$ family in [KP98, KP99].

Definition 4 *Let Ψ be a knowledge set, I be an interpretation and d be a distance between interpretations. The Max, Σ, GMax distances are defined respectively by:*

- $d_{d,Max}(I, \Psi) = \max_{\varphi \in \Psi} d(I, \varphi)$
- $d_{d,\Sigma}(I, \Psi) = \sum_{\varphi \in \Psi} d(I, \varphi)$
- *Suppose $\Psi = \{\varphi_1 \dots \varphi_n\}$. For each interpretation I we build the list $(d_1^I \dots d_n^I)$ of distances between this interpretation and the n knowledge bases in Ψ, i.e. $d_j^I = d(I, \varphi_j)$. Let $d_{d,GMax}(I, \Psi)$ be the list obtained from $(d_1^I \dots d_n^I)$ by sorting it in descending order* [4].

So, let $f \in \{Max, \Sigma, GMax\}$, such a distance induces a pre-order on interpretations:

$$I \leq_\Psi^{d,f} J \text{ iff } d_{d,f}(I, \Psi) \leq d_{d,f}(J, \Psi)$$

And the corresponding merging operator is defined by:

$$mod(\Delta_\mu^{d,f}(\Psi)) = \min(mod(\mu), \leq_\Psi^{d,f})$$

Those operators satisfy the following properties:

Theorem 2 *$\Delta^{d,Max}$ operators satisfy (IC1-IC5), (IC7), (IC8) and (Arb). $\Delta^{d,GMax}$ operators are arbitration operators. $\Delta^{d,\Sigma}$ operators are majority operators.*

It is possible to generalise the $\Delta^{d,\Sigma}$ family in the following Δ^{d,Σ^n} operators:

Definition 5 $d_{d,\Sigma^n}(I, \Psi) = \sum_{\varphi \in \Psi} d(I, \varphi)^n$.

Then the corresponding pre-order is:

$$I \leq_\Psi^{d,\Sigma^n} J \text{ iff } d_{d,\Sigma^n}(I, \Psi) \leq d_{d,\Sigma^n}(J, \Psi)$$

And the Δ^{d,Σ^n} operator is defined by:

$$mod(\Delta_\mu^{d,\Sigma^n}(\Psi)) = \min(mod(\mu), \leq_\Psi^{d,\Sigma^n})$$

It is easy to show then that:

Theorem 3 *Δ^{d,Σ^n} operators are majority operators.*

Now we illustrate the behaviour of these families on an example:

Example 3 *At a meeting of a block of flats co-owners, the chairman proposes for the coming year the construction of a swimming-pool, a tennis-court and a private-car-park. But if two of these three items are build, the rent will increase significantly. We will*

[3] Remark that the triangular inequality $d(I, J) \leq d(I, J') + d(J', J)$ is not required.

[4] the d_{GMax} distance do not strictly obey to the requirements of a distance, since it does not give numbers. In fact there is a natural mapping: choose a sufficiently big number N (where sufficiently means strictly bigger than all possible distances $d(I, \varphi_i)$, it is always possible since we work in the finite case), and then define $d_{d,GMax} = \Sigma_{j=1\dots n}(d_{i_j}^I * N^{n-j})$, where i_j denotes the jth element in the sorted list.

	φ_1	φ_2	φ_3	φ_4	$\mathbf{dist}_{Max}$	$\mathbf{dist}_{\Sigma}$	$\mathbf{dist}_{GMax}$	$\mathbf{dist}_{\Sigma^2}$
(0,0,0,0)	3	3	0	2	3	8	(3,3,2,0)	22
(0,0,0,1)	3	3	1	3	3	10	(3,3,3,1)	28
(0,0,1,0)	2	2	1	1	**2**	6	**(2,2,1,1)**	**10**
(0,0,1,1)	2	2	2	2	**2**	8	(2,2,2,2)	16
(0,1,0,0)	2	2	1	1	**2**	6	**(2,2,1,1)**	**10**
(0,1,0,1)	2	2	2	2	**2**	8	(2,2,2,2)	16
(0,1,1,0)	1	1	2	0	2	4	(2,1,1,0)	6
(0,1,1,1)	1	1	3	1	3	6	(3,1,1,1)	12
(1,0,0,0)	2	2	1	2	2	7	(2,2,2,1)	13
(1,0,0,1)	2	2	2	3	3	9	(3,2,2,2)	21
(1,0,1,0)	1	1	2	1	2	5	(2,1,1,1)	7
(1,0,1,1)	1	1	3	2	3	7	(3,2,1,1)	15
(1,1,0,0)	1	1	2	1	2	5	(2,1,1,1)	7
(1,1,0,1)	1	1	3	2	3	7	(3,2,1,1)	15
(1,1,1,0)	0	0	3	0	3	3	(3,0,0,0)	9
(1,1,1,1)	0	0	4	1	4	**5**	(4,1,0,0)	17

Table 1: Distances

denote by S, T, P respectively the construction of the swimming-pool, the tennis-court and the private-car-park. We will denote I the rent increase. The chairman outlines that build two items or more will have an important impact on the rent:

$$\mu = ((S \wedge T) \vee (S \wedge P) \vee (T \wedge P)) \rightarrow I$$

There is four co-owners $\Psi = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$. Two of the co-owners want to build the three items and don't care about the rent increase: $\varphi_1 = \varphi_2 = S \wedge T \wedge P$. The third one thinks that build any item will cause at some time an increase of the rent and want to pay the lowest rent so he is opposed to any construction: $\varphi_3 = \neg S \wedge \neg T \wedge \neg P \wedge \neg I$. The last one thinks that the flat really needs a tennis-court and a private-car-park but don't want a high rent increase: $\varphi_4 = T \wedge P \wedge \neg I$.

The propositional letters S, T, P, I will be considered in that order for the valuations:

$$mod(\mu) = \mathcal{W} \setminus \{ (0,1,1,0), (1,0,1,0), (1,1,0,0), (1,1,1,0)\}$$
$$mod(\varphi_1) = \{(1,1,1,1), (1,1,1,0)\}$$
$$mod(\varphi_2) = \{(1,1,1,1), (1,1,1,0)\}$$
$$mod(\varphi_3) = \{(0,0,0,0)\}$$
$$mod(\varphi_4) = \{(1,1,1,0), (0,1,1,0)\}$$

We sum up the calculations in table 1. The lines shadowed correspond to the interpretations rejected by the integrity constraints. Thus the result has to be found among the interpretations that are not shadowed.

With the $\Delta^{d_H,Max}$ operator, the minimum distance is 2 and the chosen interpretations are $mod(\Delta_\mu^{d_H,Max}(\Psi)) = \{(0,0,1,0), (0,0,1,1), (0,1,0,0), (0,1,0,1), (1,0,0,0)\}$. So the decision that best fit the group wishes is then not to increase the rent and to build one of the three items, or to increase the rent and build either the tennis court or the private car-park.

We can see on that example why $\Delta^{d,Max}$ operators are not IC merging operators. For example, the two interpretations $(0,0,1,0)$ and $(0,0,1,1)$ are chosen by $\Delta^{d_H,Max}$, although $(0,0,1,0)$ is better for φ_3 and φ_4 than $(0,0,1,1)$, whereas these two interpretations are equally preferred by φ_1 and φ_2. It seems then natural to globally prefer $(0,0,1,0)$ to $(0,0,1,1)$. It is in fact what demands (IC6).

The $\Delta^{d,GMax}$ family has been build with that idea of being more selective than the $\Delta^{d,Max}$ family. With the $\Delta^{d_H,GMax}$ operator the result is $mod(\Delta_\mu^{d_H,GMax}(\Psi)) = \{(0,0,1,0), (0,1,0,0)\}$, so the decision in this case will be to build either the tennis court or the car-park but without increasing the rent.

But if one chooses $\Delta^{d_H,\Sigma}$ for solving the conflict according to majority wishes, the result is then $mod(\Delta_\mu^{d_H,\Sigma}(\Psi)) = \{(1,1,1,1)\}$, and the decision will be to build the three items and to increase the rent.

Majority voting, *à la* $\Delta^{d,\Sigma}$, often seems more democratic than the other methods but, for example in this case, this only works if φ_3 accept to obey to this decision that is strictly opposed to its opinion. If φ_3 decides not to pay the rent increase, the works will perhaps not carry on because of a lack of money. So if a decision requires the approval of all the members a more consensual, arbitration like, method seems more

adequate. These kind of issues are highly related with social choice theory [Arr63, Kel78, Mou88].

On this example, one can illustrate the use of the Δ^{d,Σ^n} family, since with the operator Δ^{d_H,Σ^2} we can see that the result (on this example) is the same as with the $\Delta^{d_H,GMax}$ operator. The reason is that the power used in the definition of the operator allows to be more consensual while keeping the majority behaviour.

4 Arbitration versus Majority

We show in this section that some operators are both majority and arbitration operators. We first show that with an (over)simple operator. Then, we show that a whole family of full sense operators (the Δ^{d,Σ^n} operators) satisfy also this condition.

4.1 Drastic Distance

The simplest distance between interpretations one can define is the following one:

$$d_{Dra}(I,J) = \begin{cases} 0 & \text{if } I = J \\ 1 & \text{otherwise} \end{cases}$$

The induced distance between an interpretation and a knowledge base is then also 0 or 1 if the interpretation respectively satisfy or not the knowledge base.

It is then easy to show that the operators given with this distance by the two families $\Delta^{d,GMax}$ and $\Delta^{d,\Sigma}$ are the same. And we have the following result:

Theorem 4 *The operator $\Delta^{d_{Dra},\Sigma} = \Delta^{d_{Dra},GMax}$ satisfies (IC0)-(IC8), (Maj) and (Arb).*

This easy to state result (consequence of theorem 2) is not very surprising. But the real question is to know if more elaborate distances can lead to such *"collision"* between majority and arbitration classes. We answer this question in the next section.

4.2 Graphical study

We show in this section that some Δ^{d,Σ^n} operators are simultaneously majority and arbitration operators. For an easy explanation, we will use a graphical construction showing the behaviour of the operators "at work". In order to have a 2D representation we will restrict ourselves to two knowledge bases.

The graphical construction is simple. We put the interpretations in the plane with their distance to the φ_2 base as abscissa and with their distance to φ_1 as ordinate. Then, the aim of the merging is to find the set of interpretations that are the closest to the (0,0) point. The differences between the operators lie in the chosen distance and in this definition of "closeness".

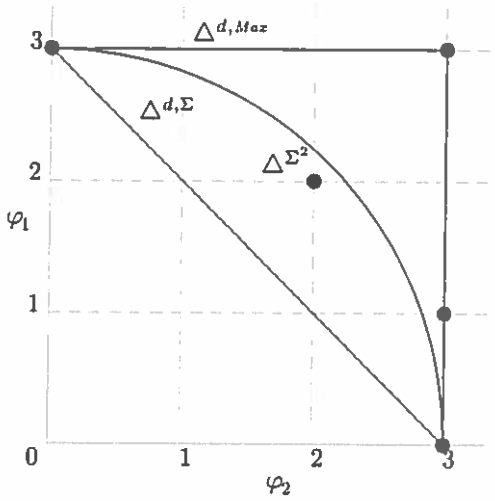


Figure 2: Merging of two knowledge bases

On figure 2, the curves represent the interpretations that are at a distance 3 from the knowledge set $\{\varphi_1, \varphi_2\}$ according to the operators $\Delta^{d,Max}$, $\Delta^{d,\Sigma}$ and Δ^{d,Σ^2}. $\Delta^{d,Max}$ is represented by a square of size a, $\Delta^{d,\Sigma}$ by the line $x = a - y$, and Δ^{d,Σ^2} by a circle arc of radius $\sqrt{a}$, where a denotes the distance from the knowledge set. The $\Delta^{d,GMax}$ operator is hardly representable in this way, but one can figure out a curve that follows the one of $\Delta^{d,Max}$ but that prefers the interpretations that are closest to the axes. We will see soon how to approximate graphically the $\Delta^{d,GMax}$ operator. Then the result of the merging, using these three operators, is the set of interpretations that the respective curves meet first when a varies from 0 to ∞.

In particular, on this example, the result for $\Delta^{d,Max}$ and Δ^{d,Σ^2} is the interpretation placed in (2,2). And for $\Delta^{d,\Sigma}$ the result is the interpretations placed in (3,0) and (0,3). In the same way, one can rebuild the preorders $\leq_\Psi^{d,Max}$, $\leq_\Psi^{d,\Sigma}$ and $\leq_\Psi^{d,\Sigma^2}$ when one consider the order the interpretations are met by the curves (when a varies from 0 to ∞).

On the figure, we can see once again the problem of $\Delta^{d,Max}$, that do not make any distinction between the (3,0) and (3,3) points for example. It is why $\Delta^{d,Max}$ is not an IC merging operator.

On the other side, $\Delta^{d,\Sigma}$ do not make any distinction on the sources of disagreements. It is absolutely not consensual, since it allows to choose interpretations

that satisfy completely one of the two bases and that dissatisfy completely the other one (the one placed in (0,3) for example), even if there are more consensual choices (an interpretation placed in (2,1) or in (2,2) for example). This behaviour can seem normal for a majority operator. But it is not systematic. Indeed, the operators Δ^{d,Σ^n} with $n > 1$ prefer more consensual choices, that is the ones closest to the line $x = y$. So, an interpretation placed in (2,2) would be prefer to one placed in (3,0).

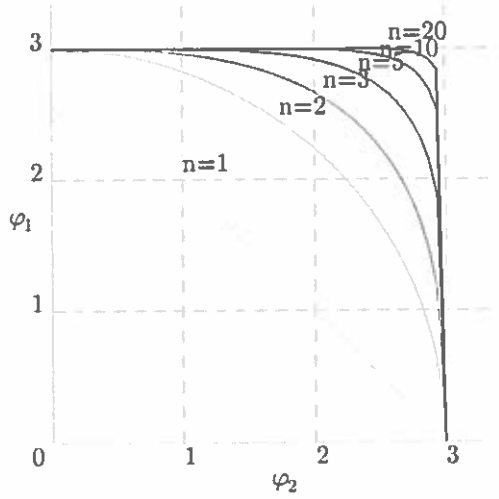


Figure 3: The Δ^{d,Σ^n} familly

Remark that the operator Δ^{d,Σ^2} is a particular operator of the Δ^{d,Σ^n} class, since it uses the Euclidean distance as distance between an interpretation and the knowledge set. This give a spherical distance, that is very natural and that obey to majority wishes but without the excesses of $\Delta^{d,\Sigma}$.

Furthermore, one can remark on figure 3 that, when one increases the value of n, the curve of Δ^{d,Σ^n} comes near to the one of $\Delta^{d,Max}$. And, with a sufficiently big n one can take the curve of Δ^{d,Σ^n} as an approximation of the one of $\Delta^{d,Max}$. But, for all n, an interpretation placed in (x, y) will always be preferred to an interpretation placed in $(x, y + 1)$ or in $(x + 1, y)$. And in fact, this way of following the $\Delta^{d,Max}$ curve but with a preference for the interpretations closest to the axes, is the one of the $\Delta^{d,GMax}$ curve. So, from some given n, we get $\Delta^{d,\Sigma^n} = \Delta^{d,GMax}$. More formally, we have the following result:

Theorem 5 *let Ψ be a knowledge set, $\exists n_0$ such that $\forall n > n_0$*

$$\Delta^{d,\Sigma^n}(\Psi) = \Delta^{d,GMax}(\Psi)$$

This result is an other answer to the partition between majority and arbitration operators. Since Δ^{d,Σ^n} operators, for all n greater than a given n_0 fixed by the maximum distance between an interpretation and a knowledge base, are both arbitration and majority operators. So the intersection between these two classes is not empty and it is possible, in a sense, to go continuously from one to the other.

5 Discussion

A possible conclusion that one can draw from the previous results is that maybe the (Arb) postulate do not capture exactly what we intend by *arbitration*. We hope we have sufficiently advocated in favour of (Arb), but let's see if we can find some alternatives.

Sometimes people take the following postulate as an expression of the arbitration behaviour (see *e.g.* [Mey01]):

(MI) $\forall n \ \Delta_\mu(\Psi_1 \sqcup \Psi_2{}^n) \leftrightarrow \Delta_\mu(\Psi_1 \sqcup \Psi_2)$

First we have to say that this postulate is not consistent with those of IC merging operators (see [KP02]). And, independently, we do not think that this postulate expresses the idea of being as close as possible to the wishes of each member of the group. It simply says that repetitions do not matter for the result of the merging (that is knowledge sets are no longer considered as multi-sets, but as simple sets). For example if the agents disagree only on the value of one propositional variable (for example consider Example 1, where the choice is simply to go to the cinema or not), then it is not possible to find compensations when building the result. So it seems sensible to take into account repetitions (that is different from being a majority operator).

An other way of thinking about arbitration operator is to allow the members to put a *right of veto* on some choices. The idea is to express the fact that the worst possible choices of each member of the group will not be chosen in the result (if possible). This can be express in this way :

(ArbV) If $\forall \varphi_i \ \exists \mu_i \ \forall \mu \ \Delta_{\mu_i \vee \mu}(\varphi_i) \vdash \mu$
and $\forall i \ \mu' \wedge \mu_i \vdash \bot$,
then $\Delta_{\mu' \vee \bigvee \mu_i}(\sqcup \varphi_i) \vdash \mu'$

Each μ_i corresponds to the worst possible choices of the agent φ_i. So what says this property is that if one can find an interpretation that does not belong to any μ_i, then no model from an μ_i will be in the result of the merging.

Even if this idea seems to be interesting, the property (ArbV) is really too complex. So it can not be checked directly, and it seems to be a very difficult task to find the corresponding condition on the syncretic assignment.

6 Conclusion

We have explored in this paper the frontier between two important subclasses of merging operators: arbitration and majority operators. The formers aiming to prefer consensual choices, whereas the latter referring to majority wishes.

An open question until now was to now if there is an intersection between these two classes or not. We have shown that it is the case, and that it is possible for an operator to be both an arbitration and a majority operator. Those operators seem to be a good compromise between democratic ideas lying in majority operators and consensual behaviour of arbitration operators.

We have introduced, in particular, a new family of operators (the Δ^{d,Σ^n} family), that allows to choose the *"consensual level"* of the majority operator according to the particular application needs.

An open question is to know if it is possible to characterised exactly what are the operators that belong simultaneously to the two classes.

Acknowledgements

The authors would like to thanks Jérôme Lang for helpful discussions about this work.

Appendix : Proof of Theorem 5

Proof : We will show that, given a knowledge set Ψ compound of m knowledge bases, and a distance d there exists n_0 such that $\forall n > n_0$ the two pre-orders $\leq_\Psi^{d,\Sigma^n}$ and $\leq_\Psi^{d,GMax}$ are the same. And we conclude by theorem 1.

We want to show that $I \leq_\Psi^{d,GMax} J$ iff $I \leq_\Psi^{d,\Sigma^n} J$.

(only if part) Consider two cases :

$I \simeq_\Psi^{d,GMax} J$, then the two sorted lists $(d^I_{\sigma(1)} \dots d^I_{\sigma(m)})$ and $(d^J_{\sigma(1)} \dots d^J_{\sigma(m)})$ are the same. And, for all n, the two distances $d_{d,\Sigma^n}(I,\Psi) = \Sigma_{i=1\dots m} {d^I_{\sigma(i)}}^n$ and $d_{d,\Sigma^n}(J,\Psi)$ are the same. So if $I \simeq_\Psi^{d,GMax} J$, then for all n $I \simeq_\Psi^{d,\Sigma^n} J$.

$I <_\Psi^{d,GMax} J$. This mean that for the two sorted lists $(d^I_{\sigma(1)} \dots d^I_{\sigma(m)})$ and $(d^J_{\sigma(1)} \dots d^J_{\sigma(m)})$ there exists $k \leq m$ such that $\forall i < k\ d^I_{\sigma(i)} = d^J_{\sigma(i)}$ and $d^I_{\sigma(k)} < d^J_{\sigma(k)}$. Consider the worst case, where $k = 1$ and such that $(d^I_{\sigma(1)} \dots d^I_{\sigma(m)}) = (x \dots x)$ and $(d^J_{\sigma(1)} \dots d^J_{\sigma(m)}) = (y\ 0 \dots 0)$ with $x < y$. The other cases will be directly retrieved by sum properties. Then the question is to find n_0 such that $\Sigma_{i=1\dots m} {d^I_{\sigma(i)}}^{n_0} < \Sigma_{i=1\dots m} {d^J_{\sigma(i)}}^{n_0}$, that is $m.x^{n_0} < y^{n_0}$. Once again, consider the worst case : $y = x + 1$, that gives $m.x^{n_0} < (x+1)^{n_0}$. It is enough to find n_0 such that $m.x^{n_0} < x^{n_0} + n.x^{n_0-1}$. That gives $n_0 > (m-1).x$. So let's note N the maximum value given by the distance d (so $x < N$). We can take $n_0 = m.N$, where m is the cardinality of Ψ. So $\forall n > n_0 = m.N$, if $I <_\Psi^{d,GMax} J$, then $I <_\Psi^{d,\Sigma^n} J$.

(if part) We want to show that, $\forall n > n_0$, if $I \leq_\Psi^{d,\Sigma^n} J$, then $I \leq_\Psi^{d,GMax} J$. Simply remark that the contraposition is : if $I <_\Psi^{d,GMax} J$, then $I <_\Psi^{d,\Sigma^n} J$, that is what we prove in the only if part (we show this with $n = n_0 = m.N$, and then derive the result for all $n > n_0$).

□

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.

[APM00] L. Amgoud, S. Parsons, and N. Maudet. Arguments, dialogue and negociation. In *Proceedings of the Fourteenth European Conference on Artificial Intelligence (ECAI'00)*, pages 338–342, 2000.

[Arr63] K. J. Arrow. *Social choice and individual values*. Wiley, New York, second edition, 1963.

[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.

[BKMS92] C. Baral, S. Kraus, J. Minker, and V. S. Subrahmanian. Combining knowledge bases consisting of first-order theo-

ries. *Computational Intelligence*, 8(1):45–71, 1992.

[CH97] L. Cholvy and T. Hunter. Fusion in logic: a brief overview. In *Proceedings of the Fourth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'97)*, Lecture Notes in Computer Science 1244, pages 86–95, 1997.

[Cho98] L. Cholvy. Reasoning about merged information. In D. M. Gabbay and Ph. Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 3, pages 233–263. Kluwer Academic Publishers, 1998.

[Dal88] M. Dalal. Investigations into a theory of knowledge base revision: preliminary report. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'88)*, pages 475–479, 1988.

[Gär88] P. Gärdenfors. *Knowledge in flux*. MIT Press, 1988.

[Kel78] J. S. Kelly. *Arrow impossibility theorems*. Series in economic theory and mathematical economics. Academic Press, New York, 1978.

[KM91] H. Katsuno and A. O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52:263–294, 1991.

[Kon00] S. Konieczny. On the difference between merging knowledge bases and combining them. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'00)*, pages 135–144, 2000.

[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)*, Lecture Notes in Artificial Intelligence 1638, pages 233–244, 1999.

[KP02] S. Konieczny and R. Pino Pérez. Merging information under constraints: a logical framework. *Journal of Logic and Computation*, 15, 2002. to appear.

[LM99] J. Lin and A. O. Mendelzon. Knowledge base merging by majority. In *Dynamic Worlds: From the Frame Problem to Knowledge Management*. Kluwer, 1999.

[LS98] P. Liberatore and M. Schaerf. Arbitration (or how to merge knowledge bases). *IEEE Transactions on Knowledge and Data Engineering*, 10(1):76–90, 1998.

[Mey01] T. Meyer. On the semantics of combination operators. *Journal of Applied Non-Classical Logics*, 11(1-2):59–84, 2001.

[Mou88] H. Moulin. *Axioms of cooperative decision making*. Monograph of the Econometric Society. Cambridge University Press, 1988.

[Rev93] P. Z. Revesz. On the semantics of theory change: arbitration between old and new information. In *Proceedings of the 12^{th} ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Databases*, pages 71–92, 1993.

[Rev97] P. Z. Revesz. On the semantics of arbitration. *International Journal of Algebra and Computation*, 7(2):133–160, 1997.

Decision Models

CP-nets - Reasoning and Consistency Testing

Carmel Domshlak **Ronen I. Brafman**
Computer Science Department
Ben-Gurion University
Beer-Sheva 84105, Israel
{dcarmel,brafman}@cs.bgu.ac.il

Abstract

Preference elicitation and preference representation play an important role in any effort to automate decision making. Unfortunately, few tools exist for preference elicitation that allow the decision maker to conveniently structure its preferences and the decision engine to efficiently reason about these preferences. One of the only such tools is CP-nets, which use a graphical representation that is superficially similar to Bayesian networks. Despite a number of interesting applications, most of the major theoretical questions related to CP-nets remain open. In particular, the complexity of determining whether one outcome is preferred to another outcome (dominance testing) is known for tree-structured networks only; moreover, little is known about the consistency of cyclic CP-nets. In this paper we show how the complexity of dominance testing depends on the structure of the CP-net. In particular we provide a new polynomial time algorithm for polytrees. In addition, we show a class of cyclic CP-nets that is *never* consistent, while other classes on which consistency can be tested for efficiently.

1 INTRODUCTION

The ability to make decisions and to assess potential courses of action is a corner-stone of many AI applications, including expert systems, autonomous agents, decision-support systems, recommender systems, configuration software, and constrained optimization applications (Chajewska, Getoor, Norman, & Shahar, 1998; DAmbrosio & Birmingham, 1995; Domshlak, Brafman, & Shimony, 2001; Junker, 2001; La Mura & Shoham, 1999; Nquyen & Haddawy, 1999). To make good decisions, we must be able to assess and compare different alternatives. Sometimes, this comparison is performed implicitly, as in many recommender systems. However, in many cases explicit information about the decision-maker's preferences is required.

In classical decision theory and decision analysis a utility function is used to represent the decision-maker's preferences. Utility functions are a powerful form of knowledge representation. They provide a quantitative measure of the desirability of different outcomes, capture attitude toward risk, and support decision making under uncertainty. However, the process of obtaining the type of information required to generate a good utility function is involved and time-consuming and requires non-negligible effort on the part of the user. In some application, this effort is necessary and/or possible, e.g., when uncertainty plays a key role, when the stakes involved are high, and when the decision-maker and the decision analyst are able and willing to engage in the required preference elicitation process. One would expect to see such effort invested when medical or important business decisions are involved. However, there are many applications where either uncertainty is not a crucial factor, or the user cannot be engaged for a lengthy period of time (e.g., in on-line product recommendation systems), or the preference elicitation process cannot be supported by a human decision analyst and must be performed by a software system (e.g., because of replicability or mass marketing aims). In such cases, elicitation of a good utility function is not a realistic option.

When a utility function cannot be or need not be obtained, one should resort to other, more qualitative forms of preference representation. Ideally, this qualitative information should be easily obtainable from the user by non-intrusive means. That is, we should be able to generate it from natural and relatively simple statements about preferences obtained from the

user, and this elicitation process should be amenable to automation. In addition, automated reasoning with this representation should be feasible and efficient.

One relatively recent framework for preference representation that addresses these concerns is that of *Conditional Preference Networks* (CP-nets) (Boutilier, Brafman, Hoos, & Poole, 1999). CP-net is a network representation of conditional preference statements under a *ceteris paribus* (all else being equal) assumption. The semantics of the exploited local preference statements capture the classical notion of (conditional) preferential independence (Keeney & Raiffa, 1976), while the CP-net allows these statements to be organized in a precise way.

CP-nets capture relatively complicated qualitative preferential statements yet preserve compactness and provide an intuitive framework for preference elicitation. This model bears a surface similarity to Bayesian network – both utilize directed acyclic graphs where each node stands for a domain variable. During preference elicitation, for each variable v_i, the decision maker is asked to specify a set of *parent* variables $P(v_i)$ that can affect her preferences over the values of v_i. That is, given a particular value assignment to $P(v_i)$, the decision maker should be able to determine a preference order for the values of v_i, all other things being equal. If $\mathcal{V}$ stands for the set of all domain variables, then v_i and $\mathcal{V} - \{v_i, P(v_i)\}$ are conditionally preferentially independent given $P(v_i)$. This information is used to create the graph of the CP-net in which each node v_i has $P(v_i)$ as its immediate predecessors. Given this structural information, the decision maker is asked to specify her preferences over the values of v_i for each assignment on $P(v_i)$. These conditional preferences over the values of v_i are captured by a *conditional preference table* which is annotated with the node v_i in the CP-net.

In this paper, we investigate two key computational properties of the CP-net model:

1. The complexity of determining whether one outcome is more preferred than another outcome given a CP-net (referred to as *dominance testing*).
2. The consistency of cyclic CP-nets.

In (Boutilier et al., 1999) it was shown that if a CP-net is defined over a set of binary variables and its graph forms a tree, then dominance testing can be performed in time polynomial in the number of variables. The exact complexity of this problem, as well as complexity of dominance testing in more general settings remains open. In this paper we investigate CP-nets over binary variables, and show that:

- When the CP-net forms a tree then the complexity of the dominance testing is quadratic in the number of variables.
- When the CP-net forms a polytree (the induced undirected graph is acyclic) then dominance testing is polynomial in the size of the CP-net description.
- When the CP-net is singly connected (there is at most one directed path between any pair of nodes), dominance testing is NP-complete. The problem remains hard even if node indegree is bounded by 6.
- Dominance testing remains in NP if the number of alternative paths between any pair of nodes in the CP-net is polynomially bounded.

The exact complexity of dominance testing in multiply-connected CP-nets remains an open problem – it is not clear if this problem is in NP, or if it is provably intractable (EXPTIME). Here we provide some structural analysis of the preferential orderings induced by CP-nets, that, by our opinion, puts some light on this question.

A CP-net is *consistent* if for every two outcomes α and β, at most one of $\alpha \succ \beta$ and $\beta \succ \alpha$ holds. Acyclic CP-nets are always consistent. Unfortunately, in the presence of cycles, the CP-net may not specify a consistent ordering. In this paper we address consistency testing for cyclic CP-nets, an issue on which, to the best of our knowledge, no previous results exist.

The rest of the paper is organized as follows: In Section 2 we provide the necessary background on preference relations and CP-nets. In Section 3 we define the problem of dominance testing formally, and we describe our results on the relation between the structure of the CP-net and the complexity of dominance testing. In Section 4 we discuss our results on consistency of cyclic CP-nets. In Section 5 we conclude. For the proofs we refer reader to our technical report (Domshlak & Brafman, 2001).

2 BACKGROUND

2.1 PREFERENCE RELATION

Suppose that, given a set of discrete variables $\mathcal{V} = \{v_1, \ldots, v_n\}$ with the corresponding domains $\mathcal{D}(v_1), \ldots, \mathcal{D}(v_n)$, the decision maker wants to define a *preference ranking* over complete assignments on $\mathcal{V}$. Each such complete assignment on $\mathcal{V}$ can be seen as a possible outcome of the decision maker's action. A

preference ranking is a total preorder $\succeq$ over the set of outcomes: $a \succeq b$ means that outcome a is equally or more preferred to the outcome b. Clearly, direct assessment of such a preference relation is infeasible due to the exponential size of the outcome space. Fortunately, a preference function can be, at least partially, specified if it exhibits sufficient structure. We describe certain types of structure here, referring to (Keeney & Raiffa, 1976) for a detailed description of these and other structural forms.

In what follows, we denote a particular assignment of values to a set $\mathcal{X} \subseteq \mathcal{V}$ as $\overline{x}$ or $\overline{x}_i$, and the concatenation of two assignments to $\mathcal{X}$ and $\mathcal{Y}$ ($\mathcal{X} \cap \mathcal{Y} = \emptyset$) by $\overline{xy}$. A subset of variables $\mathcal{X}$ is *preferentially independent* of its complement $\mathcal{Y} = \mathcal{V} - \mathcal{X}$ iff, for all $\overline{x}_1$, $\overline{x}_2$, $\overline{y}_1$, $\overline{y}_2$ holds

$$\overline{x}_1\overline{y}_1 \succeq \overline{x}_2\overline{y}_1 \text{ iff } \overline{x}_1\overline{y}_2 \succeq \overline{x}_2\overline{y}_2$$

If this relation holds, we say that $\overline{x}_1$ is preferred to $\overline{x}_2$ *ceteris paribus* (all else being equal), i.e., preferences over the assignments of $\mathcal{X}$ do not change as other attributes vary. Analogously, let $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ be a partition of $\mathcal{V}$ into three disjoint non-empty sets. $\mathcal{X}$ and $\mathcal{Y}$ are *conditionally preferentially independent* given $\overline{z}$ iff, for all $\overline{x}_1$, $\overline{x}_2$, $\overline{y}_1$, $\overline{y}_2$ holds

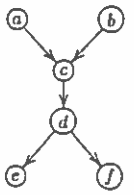
$$\overline{x}_1\overline{y}_1\overline{z} \succeq \overline{x}_2\overline{y}_1\overline{z} \text{ iff } \overline{x}_1\overline{y}_2\overline{z} \succeq \overline{x}_2\overline{y}_2\overline{z}$$

In other words, the preferential independence of $\mathcal{X}$ and $\mathcal{Y}$ only holds if $\mathcal{Z}$ is assigned to $\overline{z}$. If such a relation holds for all assignments on $\mathcal{Z}$ we say that $\mathcal{X}$ and $\mathcal{Y}$ are conditionally preferentially independent given $\mathcal{Z}$. Note that these notions are standard in multi-attribute utility theory (Keeney & Raiffa, 1976).

2.2 CP-NETS

CP-nets were introduced in (Boutilier et al., 1999) as a tool for compactly representing qualitative preference relations. This model is graphical, and it exploits conditional preferential independence in structuring a decision maker's preferences under a *ceteris paribus* assumption.

CP-nets bear a surface similarity to Bayesian networks – both utilize directed acyclic graphs where each node stands for a domain variable. During preference elicitation, for each variable v_i, the decision maker is asked to specify a set of *parent* variables $P(v_i)$ that can affect her preferences over the values of v_i. That is, given a particular value assignment to $P(v_i)$, the decision maker should be able to determine a preference order for the values of v_i, all other things being equal. Formally, v_i and $\mathcal{V} - \{v_i, P(v_i)\}$ are conditionally preferentially independent given $P(v_i)$. This information is used to create the graph of the CP-net in which each node v_i has $P(v_i)$ as its immediate predecessors. Given this structural information, the decision maker is asked to explicitly specify her preferences over the values of v_i for each assignment on $P(v_i)$. This conditional preference over the values of v_i are captured by a *conditional preference table* (CP-table) which is annotated with the node v_i in the CP-net.



$[a_1 \succ a_2]$
$[b_2 \succ b_1]$
$[(a_1 \wedge b_1) \vee (a_2 \wedge b_2) : c_1 \succ c_2;$
$(a_1 \wedge b_2) \vee (a_2 \wedge b_1) : c_2 \succ c_1]$
$[c_1 : d_1 \succ d_2;\ c_2 : d_2 \succ d_1]$
$[d_1 : e_1 \succ e_2;\ d_2 : e_2 \succ e_1]$
$[d_1 : f_1 \succ f_2 \succ f_3;\ d_2 : f_3 \succ f_1 \succ f_2]$

Figure 1: An example CP-net

Figure 1 shows an example of a CP-net with the corresponding preference tables. This network is defined over the variables $\{a, b, c, d, e, f\}$; all variables are binary except of the three-valued f. The decision maker specifies, for instance, unconditional preference over the values of a (denoted in figure by $a_1 \succ a_2$). However, if $a = a_1$ and $b = b_2$ the decision maker prefers c taking the value c_2 rather than taking the value c_1 (denoted by $(a_1 \wedge b_2) : c_2 \succ c_1$).

Now consider the CP-net above and the following three outcomes: $\alpha = a_1b_1c_1d_2e_2f_2$, $\beta = a_1b_1c_2d_2e_2f_2$, and $\gamma = a_1b_1c_2d_1e_2f_2$. The assignments α and β assign the same values to all variables except of c. In addition, given the assignment of α and β on $P(c)$, the value of c in α (c_1) is preferred to the value of c in β (c_2), all else being equal. Therefore, $\alpha \succ \beta$ is a consequence of this CP-net. In the case of β and γ, the same arguments with respect to the variable d will show that $\beta \succ \gamma$ is a consequence of this CP-net as well. Observe that $\alpha \succ \gamma$ can not be derived directly from the CP-net above. However, this relation can be inferred by the transitive closure of the direct relations $\alpha \succ \beta$ and $\beta \succ \gamma$.

Generally, nothing in these semantics forces CP-nets to be acyclic. However, the acyclicity of a CP-net ensures that it specifies a consistent preference order. That is, at most one of $\alpha \succ \beta$ and $\beta \succ \alpha$ holds for any pair of assignments α, β.

3 DOMINANCE TESTING IN CP-NETS

One of the nice properties of the CP-net model is that determining the optimal outcome is straightforward and can be done in time linear in the number of vari-

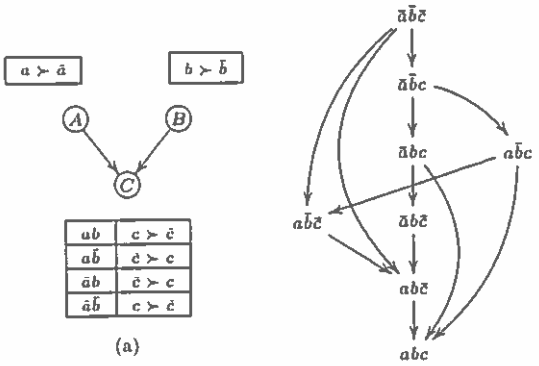


Figure 2: Illustration of a CP-net and the corresponding preference relation.

ables. This procedure is simple: traverse the variables of the given network in a topological order and set each unspecified variable to its most preferred value with respect to the values of its parents in the network. However, if some of the variables are mutually constrained by a set of hard constraints, then determining the set of Pareto optimal feasible outcomes is not trivial. In (Boutilier, Brafman, Geib, & Poole, 1997), where CP-nets were discussed for the first time, a branch and bound algorithm for determining such a set of optimal feasible outcomes, given a CP-net and a set of hard constraints on the variables of the network, was presented.

The central operation in this algorithm is *dominance testing* for a pair of outcomes: Given a CP-net $\mathcal{N}$ and two outcomes α and β, is $\alpha \succ \beta$ a consequence of $\mathcal{N}$? In other words, is the outcome α preferred to, or *dominates* β? In (Boutilier et al., 1999) this inference problem was treated as a search for a *flipping sequence* from the (purported) less preferred outcome β to the (purported) more preferred outcome α through a sequence of more preferred outcomes:

$$\beta = \gamma_0 \prec \gamma_1 \prec \cdots \prec \gamma_{m-1} \prec \gamma_m = \alpha$$

where, for $0 \leq i \leq m-1$, outcome γ_i is different from the outcome γ_{i+1} in the value of exactly one variable v_j, and $\gamma_i[j] \prec \gamma_{i+1}[j]$ given the values of $P(v_j)$ in γ_i and γ_{i+1}. Note that each value flip in such a flipping sequence is sanctioned by the network $\mathcal{N}$.

For example, consider the simple CP-net in Figure 2(a). This CP-net is defined over three variables A, B, and C; a is unconditionally preferred to $\bar{a}$, and b is unconditional preferred to $\bar{b}$, while the preference relation over the values of C is conditioned on both A and B. Figure 2(b) shows the preference relation over outcomes that this CP-net induces. The top element is the worst outcome while the bottom element is the best one. Arrows are directed from less preferred to more preferred outcomes.

Observe that, for any two outcomes α and β, every flipping sequences from β to α correspond to some unique directed path from the node β to the node α in the graph of the preference relation induced by the CP-net. For example, there are four alternative flipping sequences from the outcome $\bar{a}\bar{b}c$ to the outcome abc in the graph of the preference relation from Figure 2(b):

$$\bar{a}\bar{b}c \to a\bar{b}c \to abc$$
$$\bar{a}\bar{b}c \to a\bar{b}c \to a\bar{b}\bar{c} \to ab\bar{c} \to abc$$
$$\bar{a}\bar{b}c \to \bar{a}bc \to abc$$
$$\bar{a}\bar{b}c \to \bar{a}bc \to \bar{a}b\bar{c} \to ab\bar{c} \to abc$$

thus the CP-net in Figure 2(a) induces $abc \succ \bar{a}\bar{b}c$. In turn, there is no flipping sequence (= directed path) from $\bar{a}bc$ to $a\bar{b}\bar{c}$, thus these two outcomes are incomparable in the context of this CP-net.

A flipping sequence search algorithm was proposed in (Boutilier et al., 1999). This algorithm is deterministic and it was shown to be complete, and backtrack-free for tree-structured CP-nets with binary variables. In this section we continue the work started in (Boutilier et al., 1999), and investigate the computational properties of dominance testing between two outcomes given a CP-net. In particular we concentrate on CP-nets over binary variables, and show how by bounding the structural complexity of the network we can bound the complexity of dominance testing. In what follows, if a CP-net is defined over binary variables, it will be referred to as *binary CP-net*.

A dominance testing problem instance is represented by a three-tuple $\langle \mathcal{N}, \alpha, \beta \rangle$ in which $\mathcal{N}$ is a CP-net, α and β are the outcomes defined by the variables of $\mathcal{N}$, and the dominance hypothesis that should be tested is whether $\alpha \succ \beta$. Note that, for each variable v_i, the size of its CP-table is exponential in the number of variables in $P(v_i)$. Since the CP-tables are a part of the problem description, we can assume that for each variable, the number of its parents (its indegree in the CP-net) is bounded by a constant κ.

Another important observation is that in (Boutilier et al., 1999) dominance testing was treated and investigated as a *search* for a flipping sequence. In fact, we are interested only in *decision* whether or not such a flipping sequence exist, and in what follows we treat the dominance testing inference problem as a decision about the existence of a flipping sequence.

3.1 CP-NETS WITH RESTRICTED CONNECTIVITY

In this section, we show that when a CP-net is singly connected, then the decision about flipping sequence existence is NP-complete if $\kappa \geq 6$. We extend this result by showing that this problem remains in NP if the number of different paths between the nodes in the CP-net is polynomially bounded.

We start with a number of observations and a useful lemma. Given a dominance testing problem $\Pi = \langle \mathcal{N}, \alpha, \beta \rangle$, denote by $\mathsf{MaxFlip}(v_i)$ the maximal number of times that a variable v_i changes its value on a flipping sequence from β to α.

Observe that for any dominance testing problem over binary variables, we have that for each variable v_i in $\mathcal{V}$, $\mathsf{MaxFlip}(v_i)$ satisfies:

$$\mathsf{MaxFlip}(v_i) \leq 1 + \sum_{v_i \in P(v_j)} \mathsf{MaxFlip}(v_j) \qquad (1)$$

This follows from the fact that the variable can flip its value at most once for every possible (fixed) value of its parents. The maximal number of different values that the parents can have is bounded by the sum of the number of flips they can make plus one (for their initial configuration).

Using this relationship, we can prove that if a binary CP-net forms a singly connected DAG then, for any variable v_i, $\mathsf{MaxFlip}(v_i)$ can be bounded by n (the number of variables).

Lemma 1 *Given a dominance testing problem $\Pi = \langle \mathcal{N}, \alpha, \beta \rangle$ where $\mathcal{N}$ is a singly connected, binary CP-net, if there exists a flipping sequence from β to α then, for $1 \leq i \leq n$, where $n = |\mathcal{V}|$, we have:*

$$\mathsf{MaxFlip}(v_i) \leq n$$

Lemma 1 is crucial both for the proof of Theorem 1, and for the algorithms presented later in Section 3.3. Since every variable can flip its value at most n times, the maximal length of a flipping sequence is bounded by n^2. This implies that dominance testing for singly connected networks is in NP. In fact, we can show:

Theorem 1 *Given a dominance testing problem $\Pi = \langle \mathcal{N}, \alpha, \beta \rangle$ where $\mathcal{N}$ is a singly connected, binary CP-net, determining existence of a flipping sequence from β to α is NP-complete if $\kappa \geq 6$.*

An immediate extension of singly connected DAGs are δ-connected DAGs: a directed graph is δ-connected if the number of different paths between any two nodes in the graph is bounded by δ. By analysis similar to that of Lemma 1, we can show that, given a dominance testing problem $\Pi = \langle \mathcal{N}, \alpha, \beta \rangle$ where $\mathcal{N}$ is a δ-connected, binary CP-net, if there exist a flipping sequence from β to α then, for $1 \leq i \leq n$, where $n = |\mathcal{V}|$, we have:

$$\mathsf{MaxFlip}(v_i) \leq \delta n \qquad (2)$$

Thus:

Theorem 2 *A dominance testing problem $\Pi = \langle \mathcal{N}, \alpha, \beta \rangle$ with δ-connected CP-net $\mathcal{N}$ over binary variables is in NP if δ is polynomially bounded in the size of problem description.*

3.2 DOMINANCE TESTING FOR GENERAL BINARY CP-NETS

Theorem 1 shows that dominance testing for binary CP-nets is hard. However, the exact complexity of this problem is still an open question – it is not clear if this problem belongs to NP or it is provably intractable (EXPTIME). Here we present a graphical analysis of the preferential orderings formed by CP-nets, that, by our opinion, puts some light on this question.

An *n-dimensional hypercube* H_n is a graph on 2^n nodes each labeled by an n-bit binary number. Edges occur between nodes whose labels differ in precisely one bit. Recursively, hypercubes may be defined as follows. A 1-dimensional hypercube is an edge with one node labeled 0 and the other node labeled 1. An $(n+1)$-dimensional hypercube is constructed from two n-dimensional hypercubes, H_n^0 and H_n^1, by adding edges from each node in H_n^0 to the node in H_n^1 that has the same label and then by prefixing all of the labels in H_n^0 with a 0 and all of the labels in H_n^1 with a 1.

A directed n-dimensional hypercube $\vec{H}_n$, as it is discussed in (Baldi, 1988; Everett & Gupta, 1989; Qiu & Meijer, 1990), is obtained by an arbitrary orientation of the undirected hypercube H_n. Note that such a definition of $\vec{H}_n$ is not symmetric to the definition of H_n – the two components of H_n, H_{n-1}^0 and H_{n-1}^1, are trivially identical, while in most of the cases an arbitrary oriented $\vec{H}_n$ cannot be disassembled into two *identically oriented* components $\vec{H}_{n-1}^0$ and $\vec{H}_{n-1}^1$. In (Domshlak, 2001), *recursively directed n-dimensional hypercube* were defined as follows:

1. A 1-dimensional recursively directed hypercube is an edge with one node labeled 0 and the other node labeled 1.

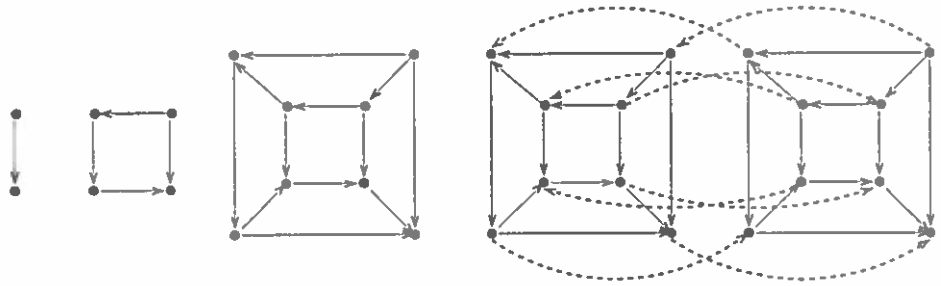


Figure 3: Construction of a recursively directed hypercube $\vec{H}_4$.

2. An $(n+1)$-dimensional recursively directed hypercube is constructed from two *identical* n-dimensional recursively directed hypercubes, $\vec{H}_n^0$ and $\vec{H}_n^1$, by adding *arbitrarily directed* edges between each node in $\vec{H}_n^0$ and the node in $\vec{H}_n^1$ that has the same label and then by prefixing all of the labels in $\vec{H}_n^0$ with a 0 and all of the labels in $\vec{H}_n^1$ with a 1.

An example of such a recursive construction is presented in Figure 3.

Recall that each node in an n-dimensional hypercube is uniquely specified by some assignment on n ordered bits. If v is a node in a recursively directed hypercube, then denote by $v|_i$ the projection of the v's label on the first i bits. An alternative view on recursively directed hypercubes was presented in (Domshlak, 2001):

A directed hypercube $\vec{H}_n$ is recursively directed if and only if there exist an ordering $x_1, \ldots, x_n$ of the label bits, and a set of boolean *orientation functions*

$$\mathcal{F} = \{\varphi_1(), \varphi_2(x_1), \ldots, \varphi_i(x_1, \ldots, x_{i-1}), \ldots \\ \ldots, \varphi_n(x_1, \ldots, x_{n-1})\}$$

such that, for each edge $(v,u) \in H_n$, if the labels of v and u disagree on x_i and $v[x_i] = 0$ $(u[x_i] = 1)$, then

$$\varphi_i(v|_{i-1}) = 0 \quad \rightarrow \quad (v,u) \in \vec{H}_n$$
$$\varphi_i(v|_{i-1}) = 1 \quad \rightarrow \quad (u,v) \in \vec{H}_n$$

Although this definition is less intuitive than the original one, the following theorem shows that it is very helpful with respect to CP-nets. First, we provide the following notation: Given a recursively directed n-dimensional hypercube $\vec{H}_n$, if m is the maximal arity of its orientation functions, then $\vec{H}_n$ is called *m-dependent*. By definition of the recursively directed hypercube, $m \leq n-1$.

Theorem 3 *Consider a partial ordering of an outcome space that is defined by n binary domain variables. This order is defined by a CP-net if and only if it forms a recursively directed, κ-dependent n-dimensional hypercube, where κ is constant.*

From Theorem 3 follows that the maximal diameter of the recursively directed, κ-dependent n-dimensional hypercubes stands for a lower bound for the complexity of flipping sequence search for general binary CP-nets. In (Domshlak, 2001) it was shown that the diameter of recursively directed n-dimensional hypercube can be exponential in n. However, this question is still open for κ-dependent recursively directed hypercubes, where κ is constant. Theorem 3 entails that answering this question will provide the answer for the NP-membership of the flipping sequence search for binary CP-nets, and vice versa.

3.3 TREE AND POLYTREE CP-NETS

As mentioned earlier, a general flipping sequence search algorithm was proposed in (Boutilier et al., 1999). This algorithm is deterministic and it was shown to be complete and backtrack-free for tree-structured CP-nets with binary variables. The decision specialization of this algorithm for the tree CP-nets is presented in Figure 4.

TreeDT performs no redundant value flips – the value of a variable is changed either in order to enable some improvement of one of its successors, or in order to achieve its value in α. Therefore, from Lemma 1, it follows that $O(n^2)$ is an upper bound on the complexity of TreeDT. Somewhat surprisingly, this is also a lower bound – in the full paper we provide an example of a dominance testing problem on a tree CP-net which requires $O(n^2)$ flips. This proves that TreeDT is an optimal algorithm.

It was already argued that TreeDT cannot be used on *polytree* CP-nets (singly-connected networks with no

TreeDT($\Pi = \langle \mathcal{N}, \alpha, \beta \rangle$)
Initialize all the variables in $\mathcal{V}$ by β.
Loop:
1. Iteratively remove all leaf variables that already assigned to their values in α.
2. If $\mathcal{V} = \emptyset$, then return *yes*.
3. Find a variable v s.t. its value can be improved, and no value of its successors in $\mathcal{N}$ can be improved, given the current assignment on $\mathcal{V}$.
4. If no such variable was found, then return *no*. Otherwise, change the value of v.

Figure 4: Flipping sequence search algorithm for tree CP-nets

undirected cycles) without backtracking, even when the variables are binary. This is due to the fact that several parents of a given node may each be allowed to have their values flipped, but only one of the choices may lead to the target outcome while the others lead to a deadend. In fact, it was believed that the problem of dominance testing in polytree CP-nets is intractable. As we now show, this problem is, in fact, polynomial, although the algorithm for its solution is non-trivial.

Theorem 4 *Dominance testing for binary, polytree CP-nets is polynomial.*

We now proceed to explain the dominance testing algorithm. The proof of its correctness appears in the full paper. Let $\Pi = \langle \mathcal{N}, \alpha, \beta \rangle$ be an instance of a dominance testing problem over a binary, polytree CP-net. We will

1. Provide a polynomial time procedure that determines $\mathsf{MaxFlip}(v_i)$ for a given variable v_i.
2. Provide a polynomial time algorithm that determines whether or not there exists a flipping sequence from β to α. This algorithm is based on a top-down execution of the computation of $\mathsf{MaxFlip}(v_i)$.

Recall that $\mathsf{MaxFlip}(v_i)$ stands for the *maximal* number of value flips of v_i that may be required by a flipping sequence from β to α. Determining $\mathsf{MaxFlip}(v_i)$, even for polytree CP-nets, requires explicit analysis of a given problem instance. First we examine the root variables of $\mathcal{N}$, then we analyzes the rest of the variables.

If $P(v_i) = \emptyset$ then the picture is relatively simple: If $\beta[i] \neq \alpha[i]$ and $\beta[i] \prec \alpha[i]$ (unconditionally) then eventually we should flip the value of v_i. However, we can do it only once, thus, in this case, $\mathsf{MaxFlip}(v_i) = 1$. Otherwise, if $\beta[i] = \alpha[i]$ then we cannot flip the value of v_i at all, and thus $\mathsf{MaxFlip}(v_i) = 0$. Finally, if $\alpha[i] \prec \beta[i]$, then we can immediately conclude that $\alpha \not\succ \beta$.

Now consider a variable v_i for which $P(v_i) = \{v_{i_1}, \ldots, v_{i_k}\} \neq \emptyset$. Observe that $\mathsf{MaxFlip}(v_i)$ depends on:

1. The values of v_i in the outcomes β and α, i.e., $\beta[i]$ and $\alpha[i]$.
2. The conditional preferences over the values of v_i, i.e., the content of its CP-table.
3. $\mathsf{MaxFlip}(v_{i_1}), \ldots, \mathsf{MaxFlip}(v_{i_k})$.
4. The actual ordering of the value flips of $P(v_i)$ on the flipping sequence from β to α.

The last point is crucial – it means that in order to determine $\mathsf{MaxFlip}(v_i)$ we should find a *particular ordering* of value flips for $P(v_i)$ that will maximize the number of possible value flips for v_i. The corresponding interleaving sequence of v_i's values, starting and finishing by $\beta[i]$ and $\alpha[i]$, respectively, will be called *maximal* and will be denoted by $\sigma(v_i)$. Clearly, $\sigma(v_i)$ consists of $\mathsf{MaxFlip}(v_i) + 1$ elements.

From Lemma 1, for $1 \leq j \leq k$, $\mathsf{MaxFlip}(v_{i_j}) \leq n$, thus the worst case for the number of different orderings of value flips of $P(v_i)$ is an order of:

$$\prod_{j=1}^{k-1} \binom{n \cdot j}{n} > n^{nk}$$

Clearly, we cannot check efficiently all these orderings in a naive manner. In what follows, we provide an algorithm DetermineMaxSequence that finds out $\sigma(v_i)$ in polynomial time. Top down execution of this algorithm on the variables of a polytree, binary CP-net, determines whether or not there exists a flipping sequence from an outcome to another outcome.

For clarity of presentation, we denote the value $\beta[i]$ of v_i by b_i and the opposite value of v_i by w_i (black/white). In what follows, we assume that the CP-table of v_i is specified in these terms. Likewise, for each variable v_i, we sequentially number the appearances of its values along $\sigma(v_i)$. For example, b_i^j stands for jth appearance of the value b_i along $\sigma(v_i)$. Thus, if $\beta[i] \neq \alpha[i]$, and $\mathsf{MaxFlip}(v_i) = 5$, then the maximal sequence $\sigma(v_i)$ will be[1]:

$$\sigma(v_i) = \mathsf{b}_i^1 \cdot \mathsf{w}_i^1 \cdot \mathsf{b}_i^2 \cdot \mathsf{w}_i^2 \cdot \mathsf{b}_i^3 \cdot \mathsf{w}_i^3$$

Informally, DetermineMaxSequence(v_i) constructs a graph (denoted as $G'_e(v_i)$) that captures *all (and only)*

[1]Note that $\mathsf{MaxFlip}(v_i)$ is odd iff $\beta[i] \neq \alpha[i]$.

feasible sequences of, up to n, value flips of v_i, where each flip is annotated with the corresponding assignment on $P(v_i)$. Although the number of the captured sequences can be exponential in n, the size of $G'_e(v_i)$ is polynomial in n. With respect to this graph, the problem of finding the maximal sequence $\sigma(v_i)$ is reduced to the problem of finding a longest path from a given node to an arbitrary other node in a directed acyclic graph (Cormen, Leiserson, & Rivest, 1990).

The graph $G'_e(v_i)$ is created in three incremental steps. At the first step, given the maximal sequences $\sigma(v_{i_1}), \ldots, \sigma(v_{i_k})$ of the parents of v_i, and the CP-table of v_i, we construct a directed graph $G(v_i)$ that captures information about all sequences of assignments on $P(v_i)$ that can enable n or less value flips of v_i. The graph $G(v_i)$ is defined as follows:

1. $G(v_i)$ consists of η nodes $\{u_1, \ldots, u_\eta\}$, where

$$\eta = \begin{cases} n, & ((n = 2j) \text{ and } (\beta[i] = \alpha[i])) \text{ or} \\ & ((n = 2j+1) \text{ and } (\beta[i] \neq \alpha[i])), \\ & j \in \mathbb{N} \\ n-1, & \text{otherwise} \end{cases}$$

   η is the maximal number of flips that can appear in the flipping sequence that starts at $\alpha[i]$ and end at $\beta[i]$.

   The nodes are labeled similarly to the elements of $\sigma(v_i)$: The node u_t is labeled by b_i^{j+1} if $t = 2j+1$, $j \in \mathbb{N}$. Otherwise, u_t is labeled by w_i^j. Observe that such a construction of $G(v_i)$ promises that the label of the last node u_η will be consistent with $\alpha[i]$.

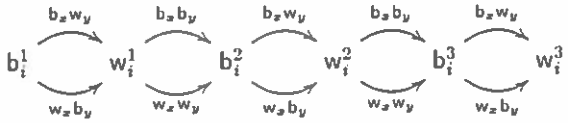
2. For $1 \leq t \leq \eta - 1$, edges from the node u_t are only to the node u_{t+1}.

3. Suppose that there are m different assignments to $P(v_i)$ under which $\mathsf{b}_i \prec \mathsf{w}_i$ ($m > 0$, since v_i depends on $P(v_i)$). In this case, for each j, there are m edges from b_i^j to w_i^j, and $2^k - m$ edges from w_i^j to b_i^{j+1} (where $2^k - m$ is the number of assignments to the parents in which $\mathsf{w}_i \prec \mathsf{b}_i$). Each edge is labeled by the corresponding assignment on $P(v_i)$. The label of the edge e is denoted by $l(e)$, and its component corresponding to the value of $v_{i_x} \in P(v_i)$ is denoted by $l(e)_{i_x}$.

For example, given a CP-net over 5 variables, consider a variable v_i with $P(v_i) = \{v_x, v_y\}$, and $\beta[i] \neq \alpha[i]$. Suppose that the computed maximal sequences for v_x and v_y are

$$\sigma(v_x) = \mathsf{b}_x^1 \cdot \mathsf{w}_x^1 \qquad \sigma(v_y) = \mathsf{b}_y^1 \cdot \mathsf{w}_y^1 \cdot \mathsf{b}_y^2 \cdot \mathsf{w}_y^2$$

and that $\mathsf{w}_i \succ \mathsf{b}_i$ if v_x and v_y are assigned either to $\mathsf{b}_x \wedge \mathsf{w}_y$, or to $\mathsf{w}_x \wedge \mathsf{b}_y$ (otherwise, $\mathsf{b}_i \succ \mathsf{w}_i$). In this case, $G(v_i)$ will be as follows:



The constructed graph $G(v_i)$ captures information about all *potentially possible* sequences of assignments on $P(v_i)$ that can enable up to n, or $n-1$ (depending on necessity) value flips of v_i. Each path that starts at the node b_i^1 uniquely corresponds to such a sequence of assignments. Although the number of these alternative sequences may be exponential in n, this graphical representation is compact: the number of edges in $G(v_i)$ is $O(n2^k)$. Note that the information about the number of times that each assignment of $P(v_i)$ can be actually achieved is not captured by $G(v_i)$. In the following we add this information and exploit it to find a maximal sequence $\sigma(v_i)$.

At the second step of construction, we expand $G(v_i)$ with respect to the maximal sequences $\sigma(v_{i_1}), \ldots, \sigma(v_{i_k})$ as follows: Each edge $e \in G(v_i)$ is replaced by a set of edges such that their labels correspond to all possible assignments of the elements of $\sigma(v_{i_1}), \ldots, \sigma(v_{i_k})$ to $l(e)$. Likewise, we add a dummy source node s_i, with an edge from s_i to the original source node of $G(v_i)$ labeled by a tuple of the first elements of $\sigma(v_{i_1}), \ldots, \sigma(v_{i_k})$. Similarly, we add a dummy target node t_i, with an edge from the original target node of $G(v_i)$ to t_i labeled by a tuple of the last elements of $\sigma(v_{i_1}), \ldots, \sigma(v_{i_k})$. We denote this extended graph by $G'(v_i)$, and Figure 3.3 illustrates $G'(v)$ for the example above.

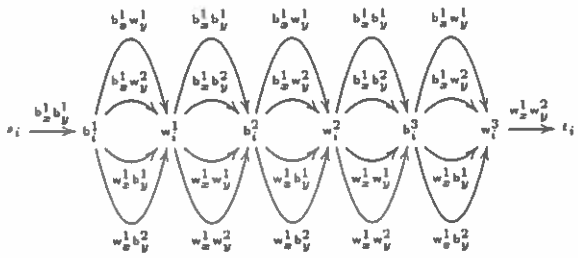


Figure 5: Example of the graph $G'(v)$.

The extended graph $G'(v_i)$ can be viewed as a projection of the maximal sequences $\sigma(v_{i_1}), \ldots, \sigma(v_{i_k})$ on the graph $G(v_i)$. Each edge in $G(v_i)$ may be replaced by $O(n^k)$ edges in $G'(v)$, and thus the number of edges in $G'(v)$ is $O(2^k n^{k+1})$.

It is easy to see that not all paths in $G'(v_i)$ starting at s_i are feasible. For example, in $G'(v_i)$ above, the assignment $\mathsf{b}_x^1\mathsf{b}_y^2$ for $P(v_i)$ cannot follow the assignment $\mathsf{b}_x^1\mathsf{w}_y^2$. Thus, now we are faced with the problem of finding a longest *feasible path* from s_i to a node in $G'(v_i)$, the label of which is consistent with $\alpha[i]$. In the following (last) step we reduce this problem to a known problem of finding a longest path in a directed acyclic graph. Let the graph $G'_e(v_i)$ have the *edges* of $G'(v_i)$ as nodes, and let its edges be defined by all allowed pairs of immediately subsequent edges in $G'(v_i)$: (e, e') is allowed if, for $1 \leq j \leq k$, either $l(e)_{v_{i_j}} = l(e')_{v_{i_j}}$ or $l(e')_{v_{i_j}}$ appears after $l(e)_{v_{i_j}}$ on $\sigma(v_{i_j})$. Such a construction is a variant of the, so called, "edge graph," which is well known in graph theory. The addition in our case is the exclusion of non-allowed edges from it. Clearly, $G'_e(v_i)$ can be constructed in time polynomial in the size of $G'(v_i)$, and the number of edges in $G'_e(v_i)$ is $O(2^{2k}n^{2k+2})$.

Figure 3.3 presents $G'_e(v_i)$ for our example. The

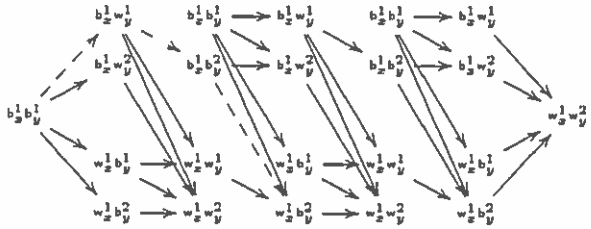


Figure 6: Example of the graph $G'_e(v)$.

dashed arrows present the longest path from the dummy source node to a node that corresponds to an assignment of $P(v_i)$ for which $\neg\alpha[i] \prec \alpha[i]$. Such a longest path in $G'_e(v_i)$ describes a maximal sequence of value flips $\sigma(v_i)$, and its length is actually $\mathsf{MaxFlip}(v_i) + 1$ nodes. In our example, $\sigma(v_i) = \mathsf{b}_i^1 \cdot \mathsf{w}_i^1 \cdot \mathsf{b}_i^2 \cdot \mathsf{w}_i^2$, and $\mathsf{MaxFlip}(v_i) = 3$. Note that if, for some i, $\beta[i] = \alpha[i]$ then the empty path in $G'_e(v_i)$ will be also acceptable since v_i may not change its value at all. In this case $\mathsf{MaxFlip}(v_i) = 0$ and $\sigma(v_i)$ will consist of only one element which corresponds to $\beta[i]$ $(= \alpha[i])$.

Procedure PolytreeDT, presented in Figure 7, summarizes the presented approach. Note that finding a set of longest paths from a node to all other nodes in a directed graph can be done in time linear in the size of the graph. Therefore, the time complexity of the call to the DetermineMaxSequence procedure with a variable v_i is bounded by the size of the constructed graph $G'_e(v_i)$ and thus is $O(2^{2k}n^{2k+2})$. PolytreeDT calls DetermineMaxSequence n times. Thus, recalling that κ is the maximal number of parents in the net, the overall complexity of the algorithm is $O(2^{2\kappa}n^{2\kappa+3})$, i.e., polynomial in the size of the corresponding CP-net.

4 CONSISTENCY TESTING FOR CYCLIC CP-NETS

As mentioned in Section 2.2, nothing in the semantics of the CP-net model forces it to be acyclic. However, the acyclicity of the network automatically provides an important property to the model: The preferential ordering over the outcomes induced by an acyclic CP-net is consistent. For cyclic CP-nets, the situation is much more complicated. For example, consider a binary cyclic CP-net over the variables $\{a, b\}$, where a depends on b and b depends on a ($\mathcal{D}(a) = \{a_1, a_2\}$ and $\mathcal{D}(b) = \{b_1, b_2\}$). If the CP-tables of a and b are specified as in Figure 8(a), then the induced preference ordering is consistent (see Figure 8(b)). However, if the CP-tables are specified as in Figure 8(c), then the specified preference ordering will not be consistent anymore (see Figure 8(d)). These examples show that the consistency of cyclic CP-nets is not guaranteed, and depends on the actual content of the CP-tables. The systematic analysis of consistency in cyclic CP-nets was left as an open problem in (Boutilier et al., 1999). Here we concentrate on cyclic binary CP-nets and discuss some new results.

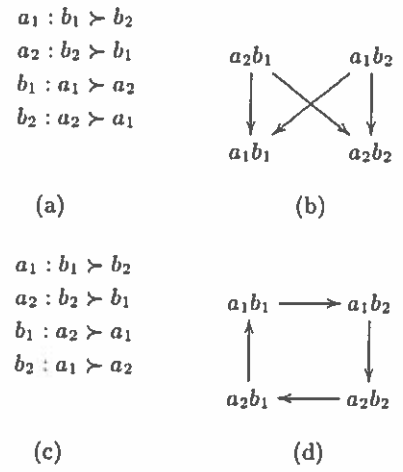


Figure 8: Examples of a consistent and inconsistent cyclic CP-nets over binary variables.

Note that in this paper we do not discuss the question of utility of cyclic CP-nets. One can argue that it is possible to cluster the variables to preserve acyclicity. Although this approach is technically feasible and probably useful in many domains, it cannot provide a

Procedure PolytreeDT ($\Pi = \langle \mathcal{N}, \alpha, \beta \rangle$)

1. Topologically sort all variables $\mathcal{V}$ based on the graph of $\mathcal{N}$.
2. For each variable $v_i \in \mathcal{V}$, call DetermineMaxSequence(Π, v_i), respecting the above ordering.
3. If one of the calls to DetermineMaxSequence returns failure, then return failure.
   Otherwise, return success.

Procedure DetermineMaxSequence(Π, v_i)

1. If $P(v_i) = \emptyset$ then
   (a) If $\alpha[i] \prec \beta[i]$ then return failure.
   (b) Otherwise, if $\beta[i] = \alpha[i]$, then $\sigma(v_i) = \mathsf{b}_i^1$, else $\sigma(v_i) = \mathsf{b}_i^1 \cdot \mathsf{w}_i^1$.
2. Otherwise, if $P(v_i) = \{v_{i_1}, \ldots, v_{i_k}\}$ then
   (a) Construct $G(v_i)$ (basing on $\beta[i]$, $\alpha[i]$, and the CP-table of v_i).
   (b) Construct $G'(v_i)$ (from $G(v_i)$, based on $\sigma(v_{i_1}), \ldots \sigma(v_{i_k})$).
   (c) Construct $G'_e(v_i)$ (from $G'(v_i)$, based on $\sigma(v_{i_1}), \ldots \sigma(v_{i_k})$).
   (d) Determine the longest path in $G'_e(v_i)$ to a node corresponding to an $\alpha[i]$-ended value flip, and derive $\sigma(v_i)$ from it.
   (e) If $\beta[i] \neq \alpha[i]$ and MaxFlip(v) = 0, then return failure. Otherwise, return success.

Figure 7: PolytreeDT algorithm

general solution. First, such a clustering will affect the space requirements of problem description crucially and thus, probably, it will degrade efficiency of reasoning about preferences. Second, in certain domains, it may be more natural to express cyclic preferences even if an acyclic representation could be used. For example, working on preference-based configuration of web page content (Domshlak et al., 2001) we found that the preferred presentation of a certain component of a web page may depend on the presentation of its neighbors in the page, whose preferred presentation depend on its presentation, etc.

Consider a binary cyclic CP-net $\mathcal{N}$ which is specified over the variables $\mathcal{V} = \{v_1, \ldots, v_n\}$. Let $C_1, \ldots, C_m$ denote the strongly connected components of $\mathcal{N}$, numbered with respect to a topological order of the CP-net. A variable v_j is a *context variable* of a strongly connected component C_i if $v_j \notin C_i$, and there exist an edge (v_j, v_k) in $\mathcal{N}$ such that $v_k \in C_i$. Finally, any complete assignment of the context variables of C_i is a *context* of C_i.

The following theorem shows that we can reduce the problem of consistency testing of cyclic CP-nets to the problem of testing the consistency of each strongly connected components in different contexts. We say that a strongly connected component C_i is *consistent* in a given context γ if the following CP-net is consistent: The nodes consist of C_i alone, and the CP-table of each node are obtained from the original CP-table by fixing the value of the context variables to γ.

Theorem 5 *A cyclic binary CP-net $\mathcal{N}$ is consistent if and only if for every strongly connected component C_i and for every context γ of C_i, we have that C_i is consistent given γ.*

Theorem 5 provides a necessary and sufficient condition for consistency of binary cyclic CP-nets, that localizes consistency testing to the strongly connected components of the network. Therefore, we can analyze only consistency testing for strongly connected CP-nets.

First, we introduce a special case of strongly connected CP-net, which we refer to as a *simple cycle*. A strongly connected CP-net is a simple cycle if there is exactly one parent for each variable in the CP-net. For example, the CP-net in Figure 9(a) forms a simple cycle, while the CP-net in Figure 9(b) forms a more complicated strongly connected structure.

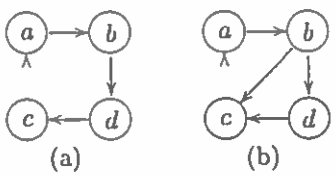


Figure 9: Illustration of the notion of simple cycle.

We assume that $\mathcal{N}$ has no redundant connections, i.e., the preference on the values of any variable is actually depends on the value of its parent. Observe that Figure 8 presented two binary, simple cycle CP-nets containing two nodes only. The CP-net in Figure 8(a-b) is consistent, while the CP-net in Figure 8(c-d) is not. The following theorem presents somewhat surprising information about the possibility that a binary, simple cycle CP-net will be consistent.

Theorem 6 *A consistent specification of a binary, simple cycle CP-net over more than 2 variables is impossible.*

In particular, Theorem 6 entails that the CP-net in Figure 8(a-b) is the *only* consistent binary CP-net that forms a simple cycle. In what follows, such a CP-net is referred to as *special-pair simple cycle*. The proof of Theorem 6 is mainly based on Lemma 2, which uses the following notation. Consider a CP-net $\mathcal{N}$ over n variables, and an assignment α on the variables of $\mathcal{N}$. We say that α *has k violations with respect to* $\mathcal{N}$, $0 \leq k \leq n$, if exactly $n-k$ variables are assigned by α to their most preferred values with respect to the assignment of α on their parents. Clearly, an optimal outcome and a worse outcome with respect to $\mathcal{N}$, has 0 and n violations, respectively.

Lemma 2 *Consider a binary, simple cycle CP-net $\mathcal{N}$ over n variables, and an assignment α on these variables, that has k violations of the preferences with respect to $\mathcal{N}$. If $0 < k < n$, then $\alpha \succ \alpha$ is a consequence of $\mathcal{N}$.*

At a first glance, Theorem 6 provides a somewhat pessimistic information about consistency of cyclic CP-nets. However, the opposite is true. We now show how Theorems 5 and 6 can be exploited to define a relatively wide class of consistent cyclic CP-nets.

Consider a CP-net $\mathcal{N}$ with strongly connected components $\{C_1, \ldots, C_m\}$, $m > 1$, such that, for $1 \leq i \leq m$, C_i forms a simple cycle. In what follows, such CP-nets are referred to as *simple-cycles-based*. An example of a simple-cycles-based CP-net graph is presented in Figure 10, where the dashed lines emphasize the strongly connected components. Theorem 7 introduces a necessary and sufficient condition for consistency of binary, simple-cycles-based CP-nets.

Theorem 7 *A binary, simple-cycles-based CP-net $\mathcal{N}$ is consistent if and only if, for each strongly connected component $C_i \in \mathcal{N}$, given any its context γ, either C_i consists of a single node, or forms a special-pair simple cycle, or C_i contains redundant dependencies, and thus is no longer cyclic.*

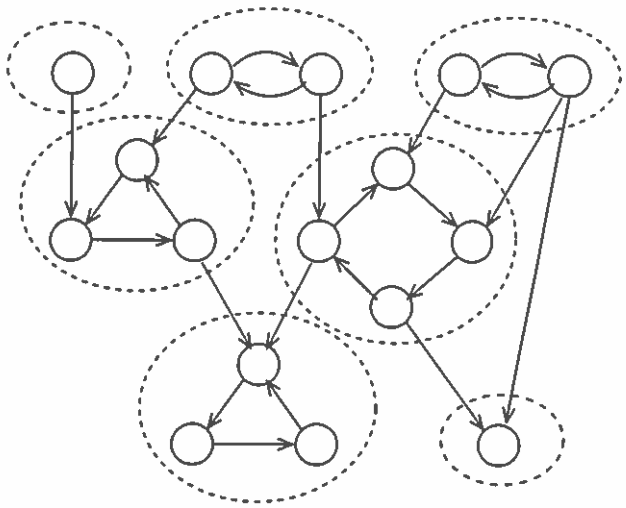


Figure 10: Graph of a simple-cycles-based CP-net.

The assumption of $m > 1$ leads to no loss of generality, since, for $m = 1$, testing consistency of $\mathcal{N}$ is trivial from Theorem 6. Similarly, observe that each "root" strongly connected component (i.e., strongly connected component with no context variables) of a CP-net from Theorem 7 either consist of a single node, or forms a special-pair simple cycle.

The immediate conclusion from Theorem 7 is that consistency testing can be performed efficiently for a wide subclass of binary simple-cycles-based CP-nets. This subclass of networks is presented by Theorem 8.

Theorem 8 *Consider a binary, simple-cycles-based CP-net $\mathcal{N}$. If, for $1 \leq i \leq m$, the number of context variables of C_i is bounded by a constant, then the consistency of $\mathcal{N}$ can be tested in polynomial time.*

The corresponding algorithm is as follows: For each strongly connected component C_i of $\mathcal{N}$, that consist of more than one node, we examine all its possible contexts, i.e., all the assignments on its context variables. Given a context γ of C_i, we project it on the CP-tables of the variables of C_i, and reduce them appropriately. If, after projecting of a context γ, C_i does not forms a special-pair simple cycle, and there are no redundant dependencies between the variables of C_i, then $\mathcal{N}$ is not consistent. Otherwise, if all the strongly connected components pass this test, then the specification of $\mathcal{N}$ is consistent.

Note that consistency testing of cyclic CP-nets re-

mains a topic for wide further research. For example, it can be shown that Lemma 2, and thus Theorem 6, does not hold neither if the variables of a simple cycle CP-net are not binary, nor if the variables are binary, but the CP-net forms more complicated strongly connected structure.

5 CONCLUSION AND FUTURE WORK

We presented results on the complexity and consistency of CP-nets. We showed how the structure of the network affects the complexity of a basic decision problem – dominance testing. Although technically quite different, our complexity results are reminiscent of similar connections established between the form of graphical structures and the complexity of the related inference problem in the areas of constraint satisfaction (Tsang, 1993) and Bayesian networks (Pearl, 1988). We showed that, perhaps surprisingly, a whole class of cyclic networks is inconsistent, regardless of the content of the tables, whereas another class can be verified in polynomial time. We are not aware of any other similar results in the area of consistency.

Our work leaves a number of interesting open questions for future research. First, a lower bound on the complexity of dominance testing in DAGs which are not singly connected needs to be established. Second, the complexity of the problem given non-binary variables remains an open problem. Finally, there is much room for work on the consistency of cyclic CP-nets. Our results, though interesting and, perhaps, surprising cover only part of the spectrum.

References

Baldi, P. (1988). Neural networks, acyclic orientations of the hypercube, and sets of orthogonal vectors. *SIAM Journal of Discrete Mathematics, 1*(1), 1–13.

Boutilier, C., Brafman, R., Geib, C., & Poole, D. (1997). A Constraint-Based Approach to Preference Elicitation and Decision Making. In *AAAI Spring Symposium on Qualitative Decision Theory*, Stanford.

Boutilier, C., Brafman, R., Hoos, H., & Poole, D. (1999). Reasoning with Conditional Ceteris Paribus Preference Statements. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 71–80.

Chajewska, U., Getoor, L., Norman, J., & Shahar, Y. (1998). Utility Elicitation as a Classification Problem. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pp. 79–88, San Francisco, CA.

Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (1990). *Introduction to Algorithms*. MIT Press.

DAmbrosio, J., & Birmingham, W. (1995). Preference-Directed Design. *Journal of Artificial Intelligence in Engineering Design, Analysis, and Manufacturing, 9*, 219–230.

Domshlak, C. (2001). On Recursively Directed Hypercubes. Tech. rep. CS-01-07, Dept. of CS, Ben-Gurion Univ.
`http://www.cs.bgu.ac.il/~dcarmel/tr0107.ps`.

Domshlak, C., & Brafman, R. (2001). CP-nets - Reasoning and Consistency Testing. Tech. rep. CS-01-06, Dept. of CS, Ben-Gurion Univ.
`http://www.cs.bgu.ac.il/~dcarmel/tr0106.ps`.

Domshlak, C., Brafman, R., & Shimony, S. E. (2001). Preference-based Configuration of Web Page Content. In *Proceedings of Seventeenth International Joint Conference on Artificial Intelligence*, pp. 1451–1456.

Everett, H., & Gupta, A. (1989). Acyclic directed hypercubes may have exponential diameter. *Information Processing Letters, 32*(5), 243–245.

Junker, U. (2001). Preference Programming for Configuration. In *Proceedings of 4th Workshop on Configuration (IJCAI-01)*, pp. 50–56.

Keeney, R. L., & Raiffa, H. (1976). *Decision with Multiple Objectives: Preferences and Value Tradeoffs*. Wiley.

La Mura, P., & Shoham, Y. (1999). Expected Utility Networks. In *Proceedings of the Fifteen Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, Stockholm, Sweden.

Nquyen, H., & Haddawy, P. (1999). The Decision-Theoretic Video Advisor. In *Proceedings of the Fifteen Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA.

Qiu, K., & Meijer, H. (1990). A note on diameter of acyclic directed hypercubes. *Information Processing Letters, 36*(1), 51–52.

Tsang, E. (1993). *Foundations of Constraint Satisfaction*. Academic Press.

On the Limitations of Ordinal Approaches to Decision-Making

Didier Dubois
IRIT - CNRS/UPS
118 route de Narbonne
31062 Toulouse

Hélène Fargier
IRIT - CNRS/UPS
118 route de Narbonne
31062 Toulouse

Patrice Perny
LIP6 - UPMC
8 rue du capitaine Scott
75015 Paris

Abstract

The aim of this paper is to evaluate the capabilities and the limitations of purely ordinal approaches to decision under uncertainty (DMU) and Multi-Criteria Decision Making (MCDM). Results obtained in this paper are valid for the two decision problems, due to their strong mathematical similarity. In the axiomatic approach followed by this paper, ordinality is captured by an axiom requiring that two ordinally equivalent pairs of alternatives should be ranked in the same way. We prove that this requirement is basically incompatible with the axiomatic framework that underlies most numerical approaches. Ordinality is especially incompatible with the transitivity of preference. When the requirement of transitivity is relaxed to quasi-transitivity, the resulting ordinal decision rules are more elaborate than what impossibility theorems in social choice suggest. Yet they prove to have a rather limited expressive power. Some ways out of this difficulty are suggested at the end of the paper.

1 INTRODUCTION

In the last ten years, decision-making has received much attention in artificial intelligence. Yet, it is a topic where scientific traditions exist, and have been around for a long time independently of AI. There are at least three decision making problems that have been studied in the past: decision making under uncertainty (DMU), social choice (SC) and multi-criteria decision making (MCDM).

In DMU, the most popular approach is numerical and relies on expected utility. This approach has been adopted in AI with the development of techniques such as influence diagrams and Markov decision processes. In social choice, the tradition which comes back to Arrow's impossibility theorem is basically ordinal. The situation is more diversified in MCDM, where both numerical approaches (multi-attribute utility theory) and ordinal approaches (outranking methods) coexist.

In artificial intelligence, the nineties have witnessed the emergence of a qualitative approach to decision making. This trend is motivated by knowledge representation concerns, namely the search for expressive languages capable of expressing preferences and reasoning with them. However, in such attempts, classical numerical decision rules are more or less given up and the question arises as to whether non numerical decision rules are rational at all and what is their expressive power.

The aim of this paper is to evaluate the capabilities and the limitations of purely ordinal decision criteria in DMU and MCDM. Results presented are valid for both decision problems, due to their mathematical similarity. We follow an axiomatic methodology, where ordinality is captured by an axiom requiring that two ordinally equivalent pairs of alternatives be ranked in the same way. We show that this requirement is basically incompatible with the axiomatic framework that underlies most numerical approaches and is especially incompatible with the transitivity of preference. In other words, this paper lays bare the wide gap existing between qualitative and numerical decision theories.

When the requirement of transitivity is relaxed to quasi-transitivity, the resulting ordinal decision rules are more elaborate than what impossibility theorems in social choice suggest. Yet they prove to have a rather limited expressive power. They are either too adventurous or not enough decisive. Some ways out of this difficulty are suggested : either by restoring some form of commensurateness forbidden by pure ordinality or, by introducing aspiration levels.

2 TWO DECISION PARADIGMS

This Section surveys the existing frameworks in decision under uncertainty and multi-criteria decision making and

shows the formal similarity between them.

2.1 DECISION MAKING UNDER UNCERTAINTY

A decision-making problem under uncertainty is a 4-tuple $(N, X, \mathcal{A}, \succsim)$, where:

- N is the set of the possible *states of nature*,
- X is the set of the possible *consequences* of acts,
- $\mathcal{A} = X^N$ is the set of *potential acts*, that is, the set of functions $x : N \to X$,
- $\succsim$ is a *preference relation* on $\mathcal{A}$, usually reflexive and transitive.

Notation For any preference relation $\succsim$, its asymmetric part defines a strict preference denoted $\succ$ and defined by: $x \succ y \Leftrightarrow (x \succsim y$ and not$(y \succsim x))$; its symmetric part defines an indifference relation denoted $\sim$ and defined by $x \sim y \Leftrightarrow (x \succsim y$ and $y \succsim x)$.

The preference relation $\succsim$ on $\mathcal{A}$ usually relies on the use of a *preference functional*, i.e. a function that maps acts to an ordered value scale. More formally, a preference function f is said to represent a preference relation $\succsim$ if and only if:

$$x \succsim y \Leftrightarrow f(x) \geq f(y)$$

Most of the preference functionals involve a numerical function u on X encoding the utility or the relative attractiveness of the consequences and a measure μ from 2^S to $[0, 1]$ representing the likelihood of the events (seen as sets of states). The latter measure must be a *capacity* on N, i.e. a mapping μ from 2^S to $[0, 1]$ such that $\mu(\emptyset) = 0, \mu(N) = 1$, and $\forall A, B \subseteq N, A \subseteq B \Rightarrow \mu(A) \leq \mu(B)$.

For instance, the seminal work of Savage [28] advocates a numerical approach, whereby uncertainty is represented by a probability function, preference is encoded by a utility functions and acts are ranked according to a particular preference functional, namely expected utility (written here in the finite setting, although Savage assumes N is infinite):

$$eu(x) = \sum_{i \in N} p(i)u(x(i)) \qquad (1)$$

Expected utility has been extended, among others, by Sarin and Wakker [27] on the basis of Choquet integral [29]. The Choquet integral is a powerful aggregation operator allowing positive and negative synergies between events by means of capacity functions other than classical probabilities (for instance, the likelihood of $A \cup B$ can be greater or less than the sum of the individual likelihoods of A and B).

Escaping the highly quantitative nature of these decision rules, more qualitative models have been proposed, that rely on ordinal evaluation scales. Among them, the pessimistic qualitative preference functional of [11, 12] is based on a utility function u from X to an ordinal scale and a possibility distribution π from S to the *same* scale. It is defined by the following preference functional :

$$v^-(x) = \min_{i \in N} \max\{1 - \pi(i), u(x(i))\} \qquad (2)$$

This criterion is actually a *weighted* extension of the Wald maximin criterion. It is a particular case of a more general criterion based on a qualitative counterpart to the Choquet integral (the Sugeno integral [31, 12]).

Recent works in AI propose to escape the use of a preference functional combining utility degrees and plausibility degrees, and they do it in two different ways. The first approach is to compare two acts on the basis of their consequences in the *most plausible states of the world* [2, 32, 7, 8, 9, 33], using more or less complex criteria. This class of approaches also aims at providing the user not only with a decision rule, but also with a high-level language (typically, a logic) for the description of the preferences, e.g. a base of conditional preferences [2] or a base of defaults encoding either desires or beliefs [33]. The criteria used in these domains are generally qualitative, e.g. criteria such as maximin, minimax regret and competitive ratio axiomatized in [7, 8, 9]. The second approach is to rely on a decision rule that compares the plausibility of the sets of states in which one act has a better consequence than the other [10, 14]. These models, that generalize Condorcet's relative majority rule, are based on pairwise comparisons. They do not compute a preference functional for each act, but start by comparing, for any pair x, y in $\mathcal{A}$ and each state i in N, the consequences $x(i)$ and $y(i)$ in order to decide whether x is at least as good as y according to this state. Then, one prefers act x to act y if and only if it is more likely to get a better consequence with x than with y. Formally, the rules of pairwise comparison (or "concordance rules") are of the form :

$$x \succsim y \Leftrightarrow C_{\succsim}(x, y) \succsim_\Lambda C_{\succsim}(y, x) \qquad (3)$$

where $C_{\succsim}(x, y) = \{i \in N, x(i) \succsim_P y(i)\}$, $\succsim_P$ is a preference relation on X, and $\succsim_\Lambda$ is a likelihood relation on 2^N. Relation $\succsim_P$ is generally a weak order on consequences (e.g. the one encoded by a utility function). Relation $\succsim_\Lambda$ can be derived from a plausibility function (i.e., a capacity μ) if any (then $A \succsim_\Lambda B \Leftrightarrow \mu(A) \geq \mu(B)$) or introduced as such by the DM. One only requires that $\succsim_\Lambda$ be a monotonic set-relation:

Definition 1 (Monotonic set-relation) *A set-relation on S is a relation $\succsim_\Lambda$ on 2^S such that $\succsim_\Lambda$ is reflexive, $\succ_\Lambda$ is transitive, $S \succ_\Lambda \emptyset$, $\forall A \subseteq N$, $(S \succsim_\Lambda A$ and $A \succsim_\Lambda \emptyset)$.*

$\succsim_\Lambda$ *is said to be* monotonic *if and only if:*
$\forall A, B, A', B'$ *such that* $A \subseteq A'$ *and* $B' \subseteq B$,
$A \succsim_\Lambda B \Rightarrow A' \succsim_\Lambda B'$.

$\succsim_\Lambda$ *is said to be* preadditive *if and only if:*
$\forall A, B, C$ *such that* $A \cap (B \cup C) = \emptyset$,
$B \succsim_\Lambda C \Leftrightarrow A \cup B \succsim_\Lambda A \cup C$.

Monotonic set-relations extend the basic properties of capacities to (possibly, partial) relations : any capacity obviously induces a monotonic set-relation: $A \succsim_\Lambda B \Leftrightarrow \mu(A) \geq \mu(B)$. Moreover, the monotonic set-relations satisfy the fundamental property of capacities ($A \subseteq B \implies A \succsim_\Lambda B$).

2.2 MULTICRITERIA DECISION-MAKING

A multi-criteria decision-making problem is again a 4-tuple $(N, \mathcal{A}, \mathcal{X} \succsim)$, where :

- $N = \{1, \ldots, n\}$ is the set of *attributes* or *criteria* describing the alternatives,
- $\mathcal{X} = \{X_1, \ldots, X_n\}$ is the set of domains of the n attributes - X_j denotes the set of possible values for attribute $j \in N$,
- $\mathcal{A}$ is the Cartesian product $\Pi_{i \in N} X_i$: it is the set of *alternatives*,
- $\succsim$ is a *preference relation* on $\mathcal{A}$.

MCDM is of the same structure as DMU, the set of criteria playing the role of states and the alternatives playing the role of acts. The main difference between MCDM and DMU is that MCDM attaches a domain X_i to each $i \in N$, although in DMU all the states $i \in N$ take their values in the same X. In other terms, a problem of DMU can be understood as a MCDM problem where $X_1 = \cdots = X_n = X$. Besides, N is always finite in MCDM. Elements of X_j are denoted x_j, and the alternative x stands for the vector $(x_1, \ldots, x_n)$.

Interestingly, there are here two distinct schools of thought in MCDM, one deriving from the DMU tradition, the other one from voting theory. The first school (see eg [20]) is essentially numerical and puts forward the aggregation of marginal utility functions $u_j(x_j)$ (measuring the attractiveness of the attribute value x_j) into a global preference functional $u(x)$ - notice that in contrast with DMU, MCDM does not use a unique utility measure; it rather uses as many marginal utilities as elements of N. One principally finds here the weighted sum, which is a counterpart to Savage's expected utility, and its generalization by the Choquet integral [18, 23]. In a more qualitative framework, one should prefer the Sugeno integral (e.g. a weighted minimum, similar to pessimistic utility). Finally, decision rules also based on aggregation functions have been proposed [24] that do not correspond to weak orders of alternatives, but can encompass weaker notions such as semi-orders, or interval orders and involve indifference thresholds; namely, the classical representation $x \succsim y \Leftrightarrow u(x) \geq u(y)$ is extended into $x \succsim y \Leftrightarrow \phi(u(x), u(y)) \geq 0$, where ϕ is a comparison function, e.g. $\phi(u(x), u(y)) = u(x) - u(y) + a$, where a is a (positive) tolerance threshold for the (weak) preference of x over y.

The second school stems from the works of Roy [25]: preferences along each criteria (whether numerical or not) are represented by an outranking relation. So-called additive concordance rules are used to perform criteria aggregation: they are based on counting the number of criteria which favor an alternative over another. This kind of rule is a particular case of a general concordance rule similar to the rule of pairwise comparison introduced in the previous Section. This general rule defines a preference relation $\succsim$ on X from the marginal preference relations $\succsim_j$ on X_j, $\forall j = 1, \ldots, n$ and a relation $\succsim_\Lambda$ on 2^N as per equation 3 where $C_{\succsim}(x, y) = \{j \in N, x_j \succsim_j y_j\}$ is now the set of criteria where x is as least as good as y and $\succsim_\Lambda$ is a monotone set-relation that encodes the relative importance of the coalitions of criteria. Again, $\succsim_\Lambda$ can be derived from an importance function or introduced as such from scratch by the decision maker.

3 ORDINALITY

In order to capture the essence of pure ordinality in DMU and MCDM we will introduce an axiom of *Ordinal Invariance*, expressing that, on each component j (attribute or state), only the relative positions of the values (of the consequences) of the two alternatives (of the two acts) are important. In the following, N is supposed to be finite, as it is the case in MCDM.

Let us first introduce the notion of compound act (or compound alternative): for any pair $x, y \in \mathcal{A}$ and for any subset of components (states or criteria) $A \subseteq N$, we can construct a mixed act/alternative xAy whose components are those of x on the elements of A, and those of y on the elements of $N \setminus A$. Formally :

$$\forall i \in N,\ (xAy)(i) = \begin{cases} x(i) & \text{if } i \in A, \\ y(i) & \text{if } i \notin A \end{cases}$$

For instance, starting from the initial relation $\succsim$, the relative position of two values on a component j is expressed by the marginal relations $\succsim_j$ defined as follows:

Definition 2 $\forall x, y \in \mathcal{A}, \forall j \in N$, $(x_j \succsim_j y_j \Leftrightarrow \forall z \in \mathcal{A}$, $(x\{j\}z) \succsim (y\{j\}z))$

There are components that never make any difference, i.e. such that $x_j \succsim_j y_j, \forall x_j, y_j \in X_j$. They are said to be *null*. More generally, one can express the conditioning of $\succsim$ on any subset $A \subseteq N$ and thus define null subsets as follows :

Definition 3 $\forall A \subseteq N, \forall x, y \in \mathcal{A} : (x \succsim y)_A \Leftrightarrow \forall z \in \mathcal{A}, xAz \succsim yAz$

Definition 4 *$A \subseteq N$ is said to be* null*if and only if* $\forall x, y \in \mathcal{A}, (x \sim y)_A$

$\succsim$ also induces a relation on the subsets of N, interpreted as the relative likelihood of events when N is a set of states, or as the relative importance of coalitions of criteria when N is a set of criteria:

Definition 5 *$\forall A, B \subseteq N$, $A \succsim_L B$ if and only if $(xAy) \succsim (xBy)$ holds for any $x, y \in X$ such that, for all non-null $\{j\} \subseteq N$, $x_j \succ_j y_j$.*

This definition is very natural: suppose that for all x and y such that x is uniformly better than y, xAy is preferred to xBy. The reason for this state of facts is obviously that the event A is more likely than the event B (or, in MCDM, that the coalition of attributes A is considered as more important than the coalition B).

Let us now state an axiom of ordinality first proposed in [14] expressing that only the relative positions of the values of the two alternatives are important, but not the values themselves, nor the positions of the two alternatives relatively to other alternatives : it means that changing the values of x and y on any component, in such a way that the order between the components is preserved, cannot change the relative preference between x and y. In other words, two ordinally equivalent pairs of alternatives should be ranked in the same way.

Definition 6 *Two pairs of alternatives (x, y) and (z, w) are said to be ordinally equivalent, denoted $(x, y) \equiv (z, w)$ if and only if :*$[\forall j \in N, (x_j \succsim_j y_j \Leftrightarrow z_j \succsim_j w_j)$ *and* $(y_j \succsim_j x_j \Leftrightarrow w_j \succsim_j z_j)]$

Axiom OI $\forall x, y, z, w \in \mathcal{A} : (x, y) \equiv (z, w) \Rightarrow (x \succsim y \Leftrightarrow z \succsim w)$.

This axiom is an adaptation to the case of weak preference relations of the non-compensation condition used in [16, 3]. It can be seen as the counterpart to an independence condition used in social choice theory [30], completed with a neutrality condition making preferences independent of the labels of the alternatives.

It is obvious that the pairwise comparison rule presented in Section 2 satisfy OI. The ordering of alternatives or acts induced by such rules is not affected by any monotonic transformation of the value scales of marginal utility functions that encode the preference according to each component, nor by any monotonic transformation of the scale encoding the importance of criteria (or, in DMU, the likelihood of events). This is why we may claim that ordinality goes along with the *non-commensurateness* of the distinct notions involved, namely the non-commensurateness of uncertainty and utility in DMU, and, in MCDM, the non-commensurateness of criteria importance and marginal utilities (and, in addition, the non-commensurateness of the different marginal scales).

Actually, axiom OI is generally not satisfied by the approaches based on a preference functional, e.g. expected utility, qualitative utility nor their counterparts in MCDM. Let us see in more details why this is so, by embedding OI into the axiomatics of Savage, since the latter justifies the use of expected utility.

4 ORDINALITY IN DMU

First, let us simplify our definitions from the fact that, in DMU, the same set of consequences is considered for each state. A first consequence of this restriction is that, among acts in $\mathcal{A}$ there are *constant acts* such that: $\exists\, a \in X : \forall\, i \in N,\ x(i) = a$. Such an act will be denoted f_a.

In Savage's framework, it seems reasonable to identify the set of constant acts and X. In this case, all the marginal preference relations $\succsim_j$ between elements of X are supposed to correspond to the global preference $\succsim_P$ derived from constant acts, provided that j is not null. This is expressed by the third axiom of Savage:

Axiom P3 $\forall A \subseteq N$, *A not null*, $\forall a, b \in X$, $\forall z \in \mathcal{A}$, $f_a Az \succsim f_b Az \Leftrightarrow a \succsim_P b$.

The DM preference relation $\succsim_P$ on X can then be soundly induced from $\succsim$ by:

$$\forall a, b \in X,\ (a \succsim_P b \Leftrightarrow f_a \succsim f_b) \qquad (4)$$

Assuming P3, Savage then defines the relative likelihood of events given by:

Definition 7 $A \succsim_L B \Leftrightarrow (\forall a, b \in X$ *such that* $a \succ_P b$, $(f_a A f_b) \succsim (f_a B f_b))$

This definition is in general not totally equivalent to definition 5 (it gives a finer $\succsim_L$). However, both definitions are equivalent when P3 and OI are assumed. Hence, relation $\succsim_L$ can be simply derived from comparisons involving only pairwise combinations of constant acts. In this context, it can be shown that OI acts as a strong version of the second and fourth axioms of Savage. Indeed, P2 (the sure thing

principle) expresses that, if x and y share the same consequence a_j on some state j (or coincide on some set of states J), then changing this (these) consequence(s) into another one (into others) will not change the global preference. OI encompasses this case, since it says that changing $x(j)$ and $y(j)$ into any pair of consequences while preserving the same preference relation as $x(j)$ and $y(j)$ (and not only the indifference induced by the equality) does not change the global preference. Formally:

Axiom P2 (Sure-Thing Principle). $\forall A \subseteq N$, $\forall x, y, z, z' \in \mathcal{A}, (xAz \succsim yAz \Leftrightarrow xAz' \succsim yAz')$

Proposition 1 *(OI + $\succsim$ reflexive) $\Rightarrow$ P2*

Moreover, the fourth Savage axiom P4 is simply an application of OI to binary acts (i.e. two acts of the form $f_a A f_b$, where $a \succ_P b$).

Axiom P4 $\forall A, B \in \mathcal{A}, \forall a, b, a', b' \in X$ such that $a \succ_P b$ and $a' \succ_P b'$, $aAb \succsim aBb \Leftrightarrow a'Ab' \succsim a'Bb'$.

Proposition 2 *(OI + $\succsim$ reflexive) $\Rightarrow$ P4*

So, ordinality is highly compatible with P2, P3, P4, and obviously with the fifth axiom of Savage (that says that $\exists a, b \in X$ such that $a \succ_P b$) or its natural strengthening P5', that says that at least three non-indifferent consequences exist :

Axiom P5' $\exists a, b, c \in X$ such that $a \succ_P b$ and $b \succ_P c$.

Major difficulties arise when P1, the first axiom of Savage, is added, which requires that $\succsim$ be a weak order :

Axiom P1 $\succsim$ is complete and transitive.

Theorem 1 *If $\succsim$ is a preference relation that satisfies P1, P3, P5' and OI , then :*

$\forall i, j \in N, \{i\} \sim_L \{j\} \Leftrightarrow i$ *and* j *are null.*

$\forall x, y \in \mathcal{A} : x \succ y \Leftrightarrow \exists i^* s.\ t. : x(i^*) \succ_P y(i^*)$ *and* $\forall j \succ_L i^*, x(j) \sim_P y(j)$

Corollary 1 *If $\succsim$ is a preference relation that satisfies P1, P3, P5' and OI , then :*

$$\exists i^* \in N : \begin{cases} \{i^*\} \succ_L N \setminus \{i^*\} \\ \forall x, y \in \mathcal{A},\ x(i^*) \succ_P y(i^*) \end{cases} \implies x \succ y$$

This theorem shows the lexicographic structure of the preference induced by P1 in a purely ordinal context and can be seen as a counterpart to Fishburn's theorem obtained in the context of multicriteria analysis [15] (similarly, Corollary 1 is a counterpart to Arrow's Theorem [1] in social choice). The resulting rule corresponds to a very particular structure of uncertainty in which all the non impossible states must be linearly ordered. Moreover, there must exist a predominant state i^*, more likely that any other state, and even more plausible than the event $N \setminus \{i^*\}$ - i.e. P1 can be satisfied if and only if one has the intuition to be in a quasi-deterministic situation. As soon as i^* gives a better consequence for x than for y, the first act is preferred. The other possible states of the world are so negligible compared to i^* that they are not considered. They only might be influent in the comparison of particular pairs (x, y) such that $x \sim_{i^*} y$.

So, ordinality is in conflict with P1. In Savage's work, P1 is justified by the goal assigned to the theory. If acts are ranked according to expected utility then the preference is reflexive, transitive and complete. But the DM Preferences could be rational without being complete nor transitive. More precisely, the reflexivity of $\succsim$ is obviously not questionable, nor the transitivity of the strict preference. But it makes sense to leave room for incomparability, apart from strict preference and indifference, so as to allow for a better description of indecisive situations: indifference between two acts x, y can be restricted to the rough similarity between the two acts whereas incomparability reflects the existence of a conflict in the comparison of x and y. So, $\sim$ is not necessarily transitive and $\succsim$ is not necessarily complete. On the other hand, it is obvious that there is no conflict between states in the comparison of constant acts. So, we may solely require the total comparability (i.e. the completeness) of constant acts. Hence we relax P1 into a weaker axiom A1:

Axiom A1 $\succsim$ is reflexive, quasi-transitive (i.e. $\succ$ is transitive) and its restriction to constant acts is complete.

However, when weakening P1, we lose important consequences of {P1, P2, P3, P4, P5}, like for instance the monotony of $\succsim$:

Definition 8 *Let $\succsim$ be a preference relation on $\mathcal{A}$ such that $a \succ_P b$ and $b \succsim_P c \Rightarrow a \succsim_P c$. $\succsim$ is said to be monotonic if and only if the two following conditions hold:*

Left Monotony (LM)*:*
if $\forall i \in A \subseteq N, x'(i) \succ_P x(i)$, *then* $x \succsim y \Rightarrow x'Ax \succsim y$

Right Monotony (RM)*:*
if $\forall i \in A \subseteq N, y(i) \succ_P y'(i)$, *then* $x \succsim y \Rightarrow x \succsim y'Ay$

This condition (denoted MO in the sequel) requires that, if x is preferred to y, then locally enhancing x or degrading y will obviously not reverse the preference. This condition obviously makes sense only in the context of a preference on consequences satisfying $a \succ_P b$ and $b \succsim_P c$ imply $a \succsim_P c$, e.g. when $\succsim_P$ is complete and quasi-transitive - this is the case when A1 holds . In Savage's framework, the latter property follows from P1 and moreover :

Proposition 3 *If $\succsim$ satisfies P1, P2 and P3 then $\succsim$ is monotonic.*

If P1 is not accepted a priori, this monotonicity property should be explicitly added to the set of axioms so as to retain as much as possible from the framework of Savage

Let us now introduce the main result of this Section, that states that ordinality not only rules out the approaches based on a preference functional (because OI is almost incompatible with P1), but it enforces the use of a rule of pairwise comparison for representing the preference on acts:

Theorem 2 *If $\succsim$ is complete on constant acts, reflexive, satisfies P3 and OI then there exists a preference relation $\succsim_P$ on X and a likelihood relation $\succsim_L$ on 2^N such that:*

$$\forall x, y \in \mathcal{A} : x \succsim y \Leftrightarrow C_{\succsim_P}(x, y) \succsim_L C_{\succsim_P}(y, x)$$

where $C_{\succsim_P}(x, y) = \{i \in N, x(i) \succsim_P y(i)\}$.

$\succsim_P$ is obviously the comparison of consequences derived from the comparison of constant acts by equation (4), and $\succsim_L$ is the likelihood relation given by equation (7). Let us focus on the latter relation. The main feature of $\succsim_L$ appears when $\succsim$ is monotonic, which is a very natural requirement, and when more than two distinct consequences exist, i.e. when P5' holds:

Proposition 4 *If $\succsim$ is complete on constant acts, reflexive, verifies P3 and OI then:*

- *$\succsim_L$ is pre-additive,*
- *If $\succsim$ is monotonic, then $\succsim_L$ is a monotonic set-relation,*
- *If $\succsim$ is quasi-transitive and satisfies P5', then $\succ_L$ satisfies the following property of* **Acceptance**: *$\forall A, B, C$ such that $A \cap B = A \cap C = B \cap C = \emptyset$, $A \cup C \succ_L B$ and $A \cup B \succ_L C \implies A \succ_L B \cup C$.*

The first property comes from the validity of the Sure Thing Principle induced by OI. The last property is a very drastic one and enforces the qualitative nature of the likelihood relation. Some important remarks are in order:

- The property of Acceptance (or "Qualitativeness") has been previously identified in [13, 17, 19] and is characteristic of the conditional likelihood relations that are compatible with deductive closure. Namely, the set of accepted beliefs A in context C such that $A \cap C \succ_L (S \setminus A) \cap C$, induced by the relation $\succ_L$, is deductively closed. In particular, this property is almost incompatible with a probabilistic representation of uncertainty [10].
- It can be shown, following [10], that any monotonic set-relation that satisfies Acceptance underlies a consequence relationship that satisfies the properties of Kraus, Lehmann and Magidor's preferential entailment [21], but for the reflexivity (the latter must be restricted to non empty events: here, nothing can be deduced from contradictory contexts). In some sense, our system of postulates is a operationally testable axiomatization of System P in terms of DMU, like Savage's axioms are an act-driven justification of Probability Theory.
- The intersection of all the events A such that $A \succ_L N \setminus A$ forms the set of most plausible states - it can be shown that is is not empty. It is the one on the basis of which almost every decision is made: when comparing x and y, the states outside this set are not taken into account, unless each of the most plausible states is indifferent among the two acts ($x(i) \sim_P y(i)$). This makes such a framework very close to the approaches that make the decision on the basis of the most plausible states only - with the slight difference that, here, the less plausible states can be considered for breaking ties.
- The above remark holds when restricting to any context $C \subseteq N$, and the approach remain valid in a dynamic setting, when new information about the actual state of the word is acquired by the DM. Decisions are dynamically consistent.

5 ORDINALITY IN MCDM

Contrary to DMU, MCDM approaches have considered ordinality as a natural assumption early in their development, and the strong opposition existing between the requirement of $\succsim$ being a weak order and ordinality (expressed by the so-called axiom of non-compensation) has been stressed years ago, by Fishburn [15]. More generally, it has been shown in [35] that the mere requirement of quasi-transitivity of $\succsim$, when added to ordinality, enforces the existence of a unique predominant set of criteria called "Oligarchy".

Definition 9 *A set $O \subseteq N$ is said to be* predominant *if and only if it is decisive in N ($\forall x, y \in \mathcal{A} : (\forall i \in O, x_i \succ_i y_i) \implies x \succ y$) and contains only vetoers (where i is a vetoer iff $\forall x, y \in \mathcal{A} : x_i \succ_i y_i \implies not(y \succ x)$).*

So, many researchers in MCDM tend to forget about non compensation, and thus focus on the study of multi-attribute decision-making using a numerical preference functional. The other school continued to stick to an ordinal approach to a large extent, through the study of pairwise comparison rules [34, 26, 4]. But, paradoxically, there are actually very few foundational works in ordinal MCDM, in comparison to multi-criteria numerical utility theory. One can however mention the pioneering work of Fishburn [16]

(see also [3, 6, 5]). The results of the previous section suggest an axiomatic framework for pairwise comparison rules, mainly deriving from OI. However, the theorem cannot be directly translated from the framework of Section 4 to MCDM, since the latter context is more general, in the sense that the domains X_i (and the related marginal comparisons $\succsim_i$) can be different from one component to the other : axiom P3 does not make sense in MCDM. Similarly, the completeness on constant acts is not meaningful as such, but should be replaced by the the minimal comparability of values within each attribute domain X_i, expressing that no conflict exists when x and y differ on a single attribute only.

Axiom MC : $\forall j \in N, \forall x, y, z, \in X : (x\{j\}z) \succsim (y\{j\}z)$ or $(y\{j\}z) \succsim (x\{j\}z)$.

MC implies the reflexivity of $\succsim$ and the completeness of the $\succsim_j$. Moreover, it is easy to conclude from MC that: $\forall j \in N, j$ not null $\Rightarrow \exists x, y \in X, x_j \succ_j y_i$.

So, we can build two alternatives x^* and x_* such that $x^*_j \succ_j x_{*j}$ if j is not null and $x^*_j = x_{*j}$ if j is null. By OI, it is easy to show that the relative importance of coalitions of criteria (Definition 5), can be equivalently defined from x^* and x_*:

Proposition 5 *if OI holds, then* $A \succsim_L B \Leftrightarrow (x^*Ax_*) \succsim (x^*Bx_*)$

We can now derive an MCDM counterpart to Theorem 2:

Theorem 3 *If the decision maker's preference* $\succsim$ *satisfies OI and MC then there exists* n *complete preference relations* $\succsim_1, \ldots, \succsim_n$, *defined on* $X_1, \ldots, X_n$ *respectively, and a* pre-additive *set-relation* $\succsim_L$ *on* 2^N, *such that:*

$$\forall x, y \in X, \quad x \succsim y \Leftrightarrow C_{\succsim}(x, y) \succsim_L C_{\succsim}(y, x)$$

where $C_{\succsim}(x, y) = \{j \in N, x_j \succsim_j y_j\}$

It shows that, in MCDM too, pure ordinality enforces the use of a concordance rule. Moreover, we can adapt the results pertaining to the likelihood relation in DMU, to the importance relation induced on sets of criteria. For instance, if $\succsim$ is monotonic, then $\succsim_L$ is a monotonic set-relation. More importantly, when the strict part of $\succsim$ is transitive and satisfies a discrimination axiom that expresses P5' in the context of MCDM, then $\succ_L$ again satisfies the property of Acceptance:

Axiom DI : $\forall j \in N, \exists x, y, z \in X$ such that $x_j \succ_j y_j$ and $y_j \succ_j z_j$.

Proposition 6 *If* $\succsim$ *is quasi-transitive and satisfies OI, MC and DI, then* $\succ_L$ *satisfies the property of Acceptance.*

It is well-known in MCDM and social choice that ordinality and quasi transitivity enforce the existence of a predominant set, i.e. of an oligarchy of vetoers. This is a consequence of the property of acceptance: the oligarchy is the intersection of all the A such as $A \succ_L N \setminus A$.

Theorem 4 *If* $\succsim$ *satisfies A1, MC, DI, MO and OI, then there exists one and only one coalition of predominant criteria.*

A similar result is known in Social Choice Theory [35] where such preference structures are called *oligarchic*. The larger the oligarchy, the less decisive the procedure. When all criteria have equal importance, the preference relation $\succsim$ reduces to the Pareto ordering. In the other extreme case, when the oligarchy contains a single criterion, the DM preferences can be described by a "dictatorial" decision rule.

In fact, the decision rule does not reduce to the existence of a mere oligarchy. Indeed, we know that the property of Acceptance, and thus the existence of a predominant event, is stable under conditioning.

Theorem 5 *If* $\succsim$ *satisfies A1, MC, DI, MO and OI, then, for any non-null* $N' \subseteq N$, *there exists a unique coalition* O *predominant in* N', *i.e. such that:*

- $O \subseteq N'$

- O *is decisive* in N' :
$\forall x, y \in \mathcal{A} : (\forall i \in O, x(i) \succ_i y(i)) \Rightarrow (x \succ y)_{N'}$,

- $\forall i \in O$, i *is a vetoer* in N':
$\forall x, y \in \mathcal{A} : x(i) \succ_i y(i) \Rightarrow not((y \succ x)_{N'})$.

Moreover, O *is such that:* $\forall A \subseteq N' : O \subseteq A \Leftrightarrow A \succ_L N' \setminus A$

The thrust of these result is that when two alternative share the same attribute values on a subset E of attribute, these attributes are neglected (this is a consequence of the sure thing principle), and *there is a set of predominant criteria in* $N \setminus E$ *that acts as a local oligarchy* and can discriminate between the alternatives. This implies that there exists not only one oligarchy of criteria, but also a *hierarchy* of oligarchies, a result that had never been emphasized before:

Corollary 2 *If* $\succsim$ *satisfies A1, MC, DI, MO and OI, then there exists a partition of* N *into coalitions* $O_1, \ldots, O_k, O_{k+1}$ *such that:*

$$\forall j \leq k, O_j \text{ is predominant in } N_j = N \setminus \bigcup_{i=1}^{j-1} O_i$$

Hence, whenever the first oligarchy does not make any difference between two alternatives, then the second level of the hierarchy becomes decisive and so on. This new result not only applies to MCDM, but also to the simpler framework of DMU. It means that, whenever the set of most

plausible states is indifferent between two acts, then the decision rule considers the second level of plausibility (i.e. the most plausible states in $N \setminus O_1$) and so on.

6 CONCLUSION: THE WAY OUT

Regarding the question "Is decision-making possible in a purely ordinal framework?", the results presented in this paper provide a contrasted answer:

- Our axiomatic study shows that yes, if the invariance under ordinal transformations of the value scale, the uncertainty and the importance scales is accepted as a norm for qualitative models, then the only available decision rules are pairwise comparison rules and a minimal preference transitivity requirement is enough to almost rule out probabilistic (resp. additive) representations of uncertainty in DMU (resp. of criteria importance in MCDM). The uncertainty structures laid bare in DMU are precisely those at work in non-monotonic reasoning. The existence of an oligarchy of most plausible states makes this model close to the qualitative models where the decision is made on the basis of the most plausible states only [2, 32, 7, 8, 9, 33], even if these methods do not necessarily rely on the unanimity on these states. The existence of sub-oligarchies shows that qualitative decision making is not restricted to the single use of the most plausible states: less plausible states do play a role, when two acts are indifferent on the basis of the top oligarchy.

- On the other hand, the identification of hierarchies of oligarchies show that, no, this kind of approach is not decisive enough, unless one is in a situation of quasi-certainty (of quasi-dictatorship): as soon as the ranking of consequences of two acts (resp. alternatives) on two maximally plausible states (resp. maximally important criteria) are conflicting, the acts (resp. alternatives) are incomparable. Hence, the larger the oligarchies, the less decisive the rule. But as the rule becomes more decisive, it becomes more dictatorial. While in MCDM, deciding on the most important criteria is a very common attitude, it is more debatable in DMU as it comes down to neglecting less likely, yet not impossible states of nature, where consequences may be unpleasant. The resulting decisions become adventurous. Consider for instance the omelette example of Savage (see [28] pages 13 to 15). The problem is how to make a six-egg omelette from a five-egg one. The new egg can be fresh of rotten. There are three feasible acts : break the egg in the omelette (bo); break it apart in a cup (bc); or throw it away (ta). The set of 5 consequences is given in Table 1 . Integers between parentheses indicate the ordering of consequences (the greater the number the higher the preference). It is easy to check that $C_{\succsim}(bo, bc) = C_{\succsim}(bo, ta) = C_{\succsim}(bc, ta) = \{\text{fresh}\}$ $C_{\succsim}(bc, bo) = C_{\succsim}(ta, bo) = C_{\succsim}(ta, bc) = \{\text{rotten}\}$. If fresh egg is more likely than rotten egg then the best act is clearly bo. If the decision-maker thinks the egg is rotten then the best act is ta. In case of total ignorance, the three acts are indifferent. So the decision making attitude induced by the approach is: break the egg in the omelette if you think the egg is fresh, throw it away if you think it is rotten, and do anything you like if you have no opinion . Clearly, this may result in many starving days, and garbage cans filled with lots of wasted fresh eggs, in the case when the state of the egg is usually hard to tell.

	fresh egg	rotten egg
bo	a 6 egg omelette (5)	nothing to eat (1)
bc	6 egg omelette + a cup to wash (4)	a 5 egg omelette + a cup to wash (2)
ta	5 egg omelette (3)	5 egg omelette (3)

Table 1: Savage's omelette example

So, if pure ordinal decision making is debatable except in very particular situations, is it possible to alter our initial framework in order to get more useful decision rules? A first idea is to relax quasi-transitivity into *acyclicity*. However, adapting results obtained in Social Choice Theory [30], it can easily be shown that non-compensatory decision models compatible with acyclicity then have very special features. For example, some state (resp. criterion) is necessarily an absolute vetoer, thus limiting the role of other states (resp. attributes).

Besides, it is clear that pairwise comparison rules are not the ultimate answer to qualitative decision problems. One may take into account the attitude of the DM in front of uncertainty like pessimism in the maximin criterion and their extensions. However such extensions require again commensurateness assumptions, even if a form of ordinality is preserved (as in [12], where the same ordinal scale is used for both uncertainty and utility).

One may also imagine situations where the preference between x and y depends on a third alternative z. As an example, consider the following relations:

$$x\Delta_z^+ y \Leftrightarrow C_{\succsim}(x,z) \succsim_L C_{\succsim}(y,z)$$
$$x\Delta_z^- y \Leftrightarrow C_{\succsim}(z,y) \succsim_L C_{\succsim}(z,x)$$

where $z \in \mathcal{A}$ is a reference point used to specify the aspiration levels of the DM, for each state of the nature. The idea underlying these rules is to compare acts by evalu-

ating their relative capabilities of meeting the DM aspirations or not. Another interesting proposal would be to require $x\Delta_z^+ y$ and/or $x\Delta_z^- y$ for several alternatives z in a set Z. In MCDM, this set represents typical preference profiles. In DMU, choosing $Z = \{f_x, x \in X\}$ (constant acts) yields a qualitative counterpart to the so-called *stochastic dominance* (see e.g. [22]). These proposals preserve the ordinal nature of decision evaluations, and the non-commensurateness of the involved value scales. What they give up is an independence property contained in OI, which requires that $\succsim$ be context-independent, thus prescribing that the preference pattern between two alternatives or acts does not depend on their relative positions with respect to a third one. Decision rules that relax this independence assumption deserve further investigation.

Acknowledgements

The authors wish to thank Jérôme Lang for his helpful comments on a previous version of the paper.

Appendix

Proof of Proposition 1. Consider $A \subseteq N$, $x, y, z, z' \in \mathcal{A}$. Since $\succsim$ is reflexive, so is also $\succsim_P$. It is then easy to check that $\forall i \in N, yAz(i) \succsim_P xAz(i) \Leftrightarrow yAz'(i) \succsim_P xAz'(i)$ and $\forall i \in N, xAz(i) \succsim_P yAz(i) \Leftrightarrow xAz'(i) \succsim_P yAz'(i)$. Hence $(xAz, yAz) \equiv (xAz', yAz')$ which implies $(xAz \succsim yAz \Leftrightarrow xAz' \succsim yAz')$ by OI.

Proof of Proposition 2. Let us choose arbitrary $A, B \subseteq N$ and $a, b, a', b' \in X$ such that $a \succ_P b$ and $a' \succ_P b'$. Since $\succsim$ is reflexive, so is also $\succsim_P$.
It is then easy to check that, $\forall i \in (A \cap B) \cup (\bar{A} \cap \bar{B})$: $aAb(i) \succsim_P aBb(i)$ and $a'Ab'(i) \succsim_P a'Bb'(i)$.
Moreover, $\forall i \in A \cap \bar{B}$: $aAb(i) = a$, $aBb(i) = b$, $a'Ab'(i) = a'$, $a'Bb'(i) = b'$. Since $a \succ_P b$ and $a' \succ_P b'$, we get $aAb(i) \succ_P aBb(i)$ and $a'Ab'(i) \succ_P a'Bb'(i)$.
Symmetrically, one can also check that for any $\forall i \in \bar{A} \cap B$: $aBb(i) \succ_P aAb(i)$ and $a'Bb'(i) \succ_P a'Ab'(i)$.
So, $(aAb, aBb) \equiv (a'Ab', a'Bb')$. Hence, using OI we get: $(aAb \succsim aAb \Leftrightarrow a'Ab' \succsim a'Bb')$ which establishes P4.

Proof of Theorem 1. It can be proved from Theorem 2 and Proposition 4 established later in this appendix. Indeed, they can be applied since P1, P3, P5′ and OI are assumed. Theorem 2 and the preadditivity property of $\succsim_L$ then implies that: $\forall x, y \in \mathcal{A}$, $x \succsim y \Leftrightarrow C_{\succsim}(x,y) \succsim_L C_{\succsim}(y,x) \Leftrightarrow C_{\succ}(x,y) \succsim_L C_{\succ}(y,x)$.

Two null events are obviously equiplausible. Reciprocally, suppose that $\exists i, j$ such that $\{i\} \sim_L \{j\}$ and $\{i\}$ not null. Consider a, b, c such that $a \succ_P b \succ_P c$ - they exist due to P5′ and P1 - and let us build the acts $x = b\{i\}c\{j\}c$, $y = c\{i\}a\{j\}c$ and $z = a\{i\}b\{j\}c$. $C_{\succ}(x,y) = \{i\}$, $C_{\succ}(y,x) = \{j\}$, $C_{\succ}(y,z) = \{j\}$, $C_{\succ}(z,y) = \{i\}$, $C_{\succ}(x,z) = \emptyset$, $C_{\succ}(z,x) = \{i,j\}$. $\{i\}$ not null implies $\{i,j\} \succ_L \emptyset$ thanks to proposition 4 and therefore $z \succ x$. On the other hand, $\{i\} \sim_L \{j\}$ implies $x \sim y$ and $y \sim z$, and thus, by P1 $x \sim z$, which contradicts $z \succ x$. So, $\{i\} \sim_L \{j\}$ implies $\{i\}$ null.

So, $N \setminus \{i,\ i \text{ null}\}$ forms a chain of singletons linearly ordered by $\succ_L$. We also know, by proposition 4 that $\succsim_L$ satisfies the property of Acceptance. It is easy to check that these two properties imply the following: $A \succ_L B \Leftrightarrow \exists i \in A$ such that $\{i\} \succ_L B$. Consider now a pair $(x,y) \in \mathcal{A}$ such that $x \succ y$. Since $x \succ y \Leftrightarrow C_{\succ}(x,y) \succ_L C_{\succ}(y,x)$, $x \succ y \Rightarrow \exists i \in C_{\succ}(x,y)$ such that $\{i\} \succ_L C_{\succ}(y,x)$. Let i^* be the most likely of these i: any state j more plausible than i^* is necessarily outside $C_{\succ}(x,y) \cup C_{\succ}(y,x)$. Hence, by completeness of $\succsim_P$, j belongs to $C_{\sim}(x,y)$.

Reciprocally, suppose that there exists an i^* such that $x(i^*) \succ_P y(i^*)$ and $x(j) \sim_P y(j)$ for all j such that $\{j\} \succ_L \{i^*\}$. Since the states are linearly ordered, i^* is more likely that any state of $C_{\succ}(y,x)$. So, using the property of acceptance, we get $C_{\succ}(x,y) \succ_L C_{\succ}(y,x)$, i.e. $x \succ y$.

Proof of Corollary 1 (sketch). Direct from Theorem 1, i^* being the maximally plausible state (it is unique since the states are linearly ordered). The property of acceptance implies that $i^* \succ_L S \setminus \{i^*\}$. $x(i^*) \succ_P y(i^*) \Rightarrow x \succ y$ is then deduced from $x \succsim y \Leftrightarrow C_{\succ}(x,y) \succsim_L C_{\succ}(y,x)$.

Proof of Proposition 3. We first establish the following:

Lemma 3.1 (Weak Unanimity) $\forall x, y \in \mathcal{A}, \forall A, B \subseteq N$ such that $A \cap B = \emptyset$, $((x \succsim y)_A$ and $(x \succsim y)_B) \Rightarrow (x \succsim y)_{A \cup B}$

Proof Consider x, y, A, B such that $A \cap B = \emptyset$ and that $\forall z$: $xAz \succsim yAz$ and $xBz \succsim yBz$. $P2$ allows us to write $xAxBz \succsim yAxBz$ and $yAxBz \succsim yAyBz$. By transitivity, this implies: $xAxBz \succsim yAyBz$ for any z, i.e. $(x \succsim y)_{A \cup B}$.

Then the monotony is established as follows:

Monotony: P1 ensures that $\succsim_P$ is transitive, so monotony can be soundly defined. Consider $A \subseteq N$ and $x, x' \in \mathcal{A}$ such that $\forall i \in A, x'(i) \succ_P x(i)$. By P3, we get $\forall i \in A, \forall z, x'\{i\}z \succsim x\{i\}z$ Since weak unanimity has been proved, it holds that $x'Az \succsim xAz, \forall z$. So, by P2, $x'Ax \succsim x$. When $x \succsim y$, we get $x'Ax \succsim y$ by transitivity which establishes LM. Similarly, for RM:, consider $y, y' \in X^S$ such that $\forall i \in A, y(i) \succ_P y'(i)$. Since weak unanimity holds, we get $yAz \succsim y'Az, \forall z$. So, by P2, $y \succsim y'Ay$. When $x \succsim y$, we get $x \succsim y'Ay$ by transitivity.

Proof of Theorem 2. The definition of $\succsim_P$ from $\succsim$ (equation 4) is self sufficient. When P3 and OI hold, $\succsim_L$ can also

be soundly defined from $\succsim$ using definition 7.

Suppose first that P5 holds and consider two consequences a and b such that $a \succ_P b$. To get the main result, it is sufficient to show that, whatever i, $x(i) \succsim_P y(i) \Leftrightarrow (aC_{\succsim}(x,y)b)(i) \succsim_P (aC_{\succsim}(y,x)b)(i)$. Indeed, in that case, $(x,y) \equiv (aC_{\succsim}(x,y)b, aC_{\succsim}(y,x)b)$ (recall that $\succsim_P$ is complete) and OI would enable the deduction of $x \succsim y \Leftrightarrow aC_{\succsim}(x,y)b \succsim aC_{\succsim}(y,x)b$. By definition of $\succsim_L$, $C_{\succsim}(x,y) \succsim_L C_{\succsim}(y,x) \Leftrightarrow aC_{\succsim}(x,y)b \succsim aC_{\succsim}(y,x)b$. Hence, we would get $x \succsim y \Leftrightarrow C_{\succsim}(x,y) \succsim_L C_{\succsim}(y,x)$.

Let us show that, for all i, $(x(i) \succsim_P y(i) \Leftrightarrow (aC_{\succsim}(x,y)b)(i) \succsim_P (aC_{\succsim}(y,x)b)(i))$:
- Any i such that $x(i) \sim_P y(i)$ belongs to $C_{\succsim}(x,y)$ and to $C_{\succsim}(y,x)$. Hence $(aC_{\succsim}(x,y)b)(i) = (aC_{\succsim}(y,x)b)(i) = a$. So, by reflexivity of $\succsim$ and thus of $\succsim_P$ we have: $(aC_{\succsim}(x,y)b)(i) \sim_P (aC_{\succsim}(y,x)b)(i)$.
- Any i such that $x(i) \succ_P y(i)$ belongs to $C_{\succsim}(x,y)$ and does not belong to $C_{\succsim}(y,x)$. Hence $(aC_{\succsim}(x,y)b)(i) = a$ and $(aC_{\succsim}(y,x)b)(i) = b$. Since $a \succ_P b$, $(aC_{\succsim}(x,y)b)(i) \succ_P (aC_{\succsim}(y,x)b)(i)$.
- Similarly, for i such that $y(i) \succ_P x(i)$, we get $(aC_{\succsim}(y,x)b)(i) \succ_P (aC_{\succsim}(x,y)b)(i)$.
Since $\succsim_P$ is complete, this proves that $x(i) \succsim_P y(i) \Leftrightarrow C_{\succsim}(x,y)b(i) \succsim_P aC_{\succsim}(y,x)b(i)$.

On the other hand, suppose that P5 does not hold. Since $\succsim_P$ is complete, this means that: $\forall x,y \in \mathcal{A}, \forall i \in N$, $x(i) \sim_P y(i)$. Thus $\forall x,y, C_{\succsim}(x,y) = C_{\succsim}(y,x) = S$. Thus $\forall x,y, C_{\succsim}(x,y) \succsim_L C_{\succsim}(y,x)$ (reflexivity of $\succsim$ and thus of $\succsim_L$). Moreover, by OI and reflexivity of $\succsim$ we get $x \succsim y, \forall x,y$. From $[\forall x,y, C_{\succsim}(x,y) \succsim_L C_{\succsim}(y,x)$ and $\forall x,y, x \succsim y]$, we can derive $[\forall x,y, C_{\succsim}(x,y) \succsim_L C_{\succsim}(y,x) \Leftrightarrow x \succsim y]$.

Proof of Proposition 4. Under P5, $\exists a,b \in X$ such that $a \succ_P b$. So, $A \succsim_L B \Leftrightarrow aAb \succsim_L aBb$. Moreover, OI and $\succsim$ complete on constant acts imply P2 (Proposition 1). *Preadditivity of* $\succsim_L$. Consider A, B, C such that $A \cap (B \cup C) = \emptyset$. $B \succsim_L C \Leftrightarrow aBb \succsim aCb$. Since $A \cap (B \cup C) = \emptyset$ and P2 holds, we have $aBb \succsim aCb \Leftrightarrow aBaAb \succsim aCaAb$, which means $B \succsim_L C \Leftrightarrow A \cup B \succsim_L A \cup C$.

Monotony of $\succsim_L$. Suppose that $A \succsim_L B$, i.e. $aAb \succsim aBb$. By LM, we get $aAa(\bar{A} \cap C)b \succsim aBb$ i.e. $A \cup C \succsim_L B$. Suppose that $A \succsim_L B \cup C$, i.e. $aAb \succsim a(B \cup C)b$. By RM, we get $aAb \succsim aBb$, i.e. $A \succsim_L B$.

Acceptance P5 requires that X contains at least three elements a,b,c such that $a \succ b$, $a \succ c$, $b \succ c$. So, by definition 5, $A \succsim_L B \Leftrightarrow (aAb) \succsim (aBb)$. Let $E, F, G \subseteq N$ be such that $E \cap F = E \cap G = F \cap G = \emptyset$, $E \cup G \succ_L F$, and $E \cup F \succ_L G$ and consider the six following acts: $ef = a(E \cup F)b$, $eg = a(E \cup G)b$, $fg = a(F \cup G)b$, $e = aEb$, $f = aFb$, $g = aGb$. Thanks to Theorem 2, $eg \succ f \Leftrightarrow E \cup G \succ_L F$ and $ef \succ g \Leftrightarrow E \cup F \succ_L G$, and therefore $eg \succ f$ and $ef \succ g$. Since E, F and G are disjoint, one can also consider the acts: $x = aEbFcGb$, $y = bEcFaGb$, $z = cEaFbGb$. It is easy to check that $(x,y) \equiv (ef,g)$ and $[(y,z) \equiv (eg,f)]$. Hence by OI we get $x \succ y$ and $y \succ z$. By transitivity of $\succ$, this implies that $x \succ z$. Since $(x,z) \equiv (a,bc)$, OI implies that $a \succ bc$ i.e. $aEb \succ a(F \cup G)b$. By Theorem 2, this is equivalent to $E \succ_L F \cup G$.

Proof of Proposition 5. Let $x,y \in X$ such that, for all non-null $\{j\} \subseteq N$, $x_j \succ_j y_j$. It is easy to check that $(x,y) \equiv (x^*, x_*)$ and $(xAy, xBy) \equiv (x^*Ax_*, x^*Bx_*)$. So, by OI : $(xAy) \succsim (xBy) \Leftrightarrow (x^*Ax_*) \succsim (x^*Bx_*)$. So, $\forall x,y \in X$ such that, for all non-null $\{j\} \subseteq N$, $x_j \succ_j y_j : (xAy) \succsim (xBy) \Leftrightarrow (x^*Ax_*) \succsim (x^*Bx_*)$. By definition 5, this proves $A \succsim_L B \Leftrightarrow (x^*Ax_*) \succsim (x^*Bx_*)$.

Proof of Theorem 3.
Completeness of $\succsim_j$. The proof is direct from MC. Indeed, MC means that for all $x,y \in \mathcal{A}$, and whatever $j \in N$ it holds that: $\forall z \in \mathcal{A}, (x\{j\}z) \succsim (y\{j\}z)$ or $\forall z \in \mathcal{A}, (y\{j\}z) \succsim (x\{j\}z)$. By definition (4), this means that $x_j \succsim_j y_j$ or $y_j \succsim_j x_j$. Since this holds for any pair (x,y), this proves that the $\succsim_j$, $j = 1, \ldots, n$ are complete.

Preadditivity of $\succsim_L$. Similar to the one of DMU, with x^* (resp. x_*) playing the role of the constant act a (resp. the constant act b). Indeed, P2 still holds in MCDM without any modification of its definition.

Main result. Let us now consider a pair $(v,w) \in \mathcal{A}^2$ such that $v \succsim w$. Let us denote $A = C_{\succsim}(v,w)$ and $B = C_{\succsim}(w,v)$. Consider the relation $\succsim_L$ between coalitions defined by equation (5). Since OI holds we know by Proposition 5 that $A \succsim_L B$ if and only if $(x^*Ax_*) \succsim (x^*Bx_*)$, Now, for any $j \in N$, $\succsim_j$ is complete. So, for criteria j, only 3 cases are possible:
- $v_j \succ_j w_j : (x^*Ax_*)_j = (x^*)_j$ and $(x^*Bx_*)_j = (x_*)_j$, thus $(x^*Ax_*)_j \succ_j (x^*Bx_*)_j$
- $w_j \succ_j v_j : (x^*Ax_*)_j = (x_*)_j$ and $(x^*Bx_*)_j = x^*_j$, thus $(x^*Bx_*)_j \succ_j (x^*Ax_*)_j$
- $v_j \sim_j w_j : (x^*Ax_*)_j = x^*_j$ and $(x^*Bx_*)_j = x^*_j$, thus $(x^*Ax_*)_j \sim_j (x^*Bx_*)_j$
So, $C_{\succ}(v,w) = C_{\succ}(x^*Ax_*, x^*Bx_*)$, $C_{\succ}(w,v) = C_{\succ}(x^*Bx_*, x^*Ax_*)$, $C_{\sim}(v,w) = C_{\sim}(x^*Ax_*, x^*Bx_*)$. Thus $C_{\succ}(v,w) = C_{\succsim}(x^*Ax_*, x^*Bx_*)$ and $C_{\succ}(w,v) = C_{\succsim}(x^*Bx_*, x^*Ax_*)$. So $(v,w) \equiv (x^*Ax_*, x^*Bx_*)$. By OI, $v \succsim w \Leftrightarrow (x^*Ax_*) \succsim (x^*Bx_*)$. Moreover, we know that $(x^*Ax_*) \succsim (x^*Bx_*) \Leftrightarrow A \succsim_L B$. Thus: $v \succsim w \Leftrightarrow C_{\succsim}(v,w) \succsim_L C_{\succsim}(w,v)$.

Proof of Proposition 6. The proof is similar to the one of DMU, with $x \succ y \succ z$ playing the roles of the constant act a, b and c. They indeed exist, due to DI.

Proof of Theorem 4. This theorem is an obvious consequence of Theorem 5 - just by letting $N' = N$. A direct proof can be obtained by transposing a result of Weymark [35] in Social Choice Theory to the case of a cartesian product.

Proof of Theorem 5. By Theorem 3, we know that $x \succsim y \Leftrightarrow C_{\succ}(x,y) \succsim_L C_{\succ}(y,x)$. We also proved (Proposition 6) that $\succsim_L$ is an acceptance relation. Its monotonicity can be easily derived from MO (with a proof similar to the proof of proposition 4).

Lemma 5.1 : If $\succsim$ satisfies A1, DI, MO and OI then:
$\forall O \subseteq N' \subseteq N$, O is decisive in $N' \Leftrightarrow O \succ_L N' \setminus O$

Proof: Suppose that O is decisive in N' and consider the acts $x^*Ox_*(N' \setminus O)z$ and $x^*(N' \setminus O)x_*Oz$. Since O is decisive in N', we get $x^*Ox_*(N' \setminus O)z \succ x^*(N' \setminus O)x_*Oz, \forall z$. So, it holds that $x^*Ox_*(N' \setminus O)x_* \succ x^*(N' \setminus O)x_*Ox_*$, i.e. $O \succsim_L N' \setminus O$.

Reciprocally, suppose that $O \subseteq N'$, $O \succ_L N' \setminus O$ and consider x, y such as $\forall i \in O, x(i) \succ_P y(i)$. Consider the acts $xN'z$ and $yN'z$. It holds that $O \subseteq C_{\succ}(xN'z, yN'z)$. Since $\succsim_P$ is complete, we also have: $C_{\succ}(yN'z, xN'z) \subseteq N \setminus O$ and thus $C_{\succ}(yN'z, xN'z) \subseteq N' \setminus O$. By monotony of $\succsim_L$, $O \succ_L N' \setminus O$ gives $C_{\succ}(xN'z, yN'z) \succ_L C_{\succ}(yN'z, xN'z)$. By additivity of $\succ_L$, this is equivalent to $C_{\succsim}(xN'z, yN'z) \succ_L C_{\succsim}(yN'z, xN'z)$, i.e. $xN'z \succ yN'z$. Since we did not make any assumption on z, this means $(x \succ y)_{N'}$.

Lemma 5.2 : If $\succsim$ satisfies A1, DI, MO and OI, then there exists at most one coalition predominant in N'

Proof: Suppose N' contains at least two distinct predominant coalitions O_1 and O_2.
On the one hand we get $x^*O_1x_*(N' \setminus O_1)z \succ x_*O_1x^*(N' \setminus O_1)z, \forall z$, since O_1 is decisive in N'.
On the other hand, since O_2 is predominant in N' and distinct from O_1, there exists at least one $i \in O_2 \setminus O_1$ that is a vetoer. Thus, there is at least one $i \in N' \setminus O_1$ that is a vetoer. This fact contradicts $x^*O_1x_*(N' \setminus O_1)z \succ x_*O_1x^*(N' \setminus O_1)z$.

Proof of the main result: Let us now remark that if $A \subseteq N'$ is decisive in N' and $B \subseteq N'$ is decisive in N', then $A \cap B$ is non empty and decisive in N'. Indeed, thanks to Lemma 5.1, this implies that $A \succ_L \bar{A} \cap N'$ and $B \succ_L \bar{B} \cap N'$. Thus $(A \cap B) \cup (A \cap \bar{B}) \succ_L (\bar{A} \cap \bar{B} \cap N') \cup (\bar{A} \cap B)$ and $(A \cap B) \cup (\bar{A} \cap B) \succ_L (\bar{A} \cap \bar{B} \cap N') \cup (A \cap \bar{B})$. By monotony of $\succsim_L$, we get $(A \cap B) \cup (A \cap \bar{B}) \succ_L (\bar{A} \cap \bar{B} \cap N') \cup (\bar{A} \cap B)$ and $(A \cap B) \cup (\bar{A} \cap B) \cup (\bar{A} \cap \bar{B} \cap N') \succ_L (A \cap \bar{B})$. By acceptance, we get: $(A \cap B) \succ_L (\bar{A} \cap \bar{B} \cap N') \cup (\bar{A} \cap B) \cup (A \cap \bar{B})$, i.e. $(A \cap B) \succ_L N' \setminus (A \cap B)$. So, $(A \cap B)$ is decisive in N'. It cannot be empty, otherwise we would get $\emptyset \succsim_L N'$ which contradicts the assumption that N' is not null.

Since N' is not null, $N' \succ_L \emptyset$ (applying Prop. 4). From Lemma 5.1, thus know that there exist at least one decisive coalition in N': N' itself. So, $O = \bigcap_{A, A \subseteq N', A \succ_L N' \setminus A}$, the intersection of all decisive sets in N' is a non empty subset of N' which is decisive in N'. Let us show that any $i \in O$ is a vetoer. Consider x, y such that $x(i) \succ_P y(i)$. i belongs to $C_{\succ}(xN'z, yN'z)$. Suppose that $yN'z \succ xN'z$, which is equivalent to $C_{\succ}(yN'z, xN'z) \succ_L C_{\succ}(xN'z, yN'z)$. Since $i \in C_{\succ}(xN'z, yN'z)$ and $C_{\succ}(yN'z, xN'z) \subseteq N' \setminus \{i\}$, the monotony of $\succsim_L$ implies $N' \setminus \{i\} \succ_L \{i\}$. So, i cannot belong to $O = \bigcap_{A, A \succ_L N' \setminus A}$. This yields a contradiction.So, i must be a vetoer.
$O = \bigcap_{A, A \succ_L N' \setminus A}$ is thus a set predominant in N'. By Lemma 5.2, we know that it is unique.

Proof of the additional property : By definition of O, if $A \succ_L N' \setminus A$, then $O \subseteq A$. Conversely, if $O \subseteq A$, $O \succ_L N' \setminus O$ implies by monotony of $\succsim_L$ that $A \succ_L N' \setminus A$.

Proof of Corollary 2. The existence of the top predominant set O_1 is due to Theorem 5 (just by letting $N' = N$). Now, suppose that the property holds until level j. So, there exists a subset O_j predominant in $N_j = N \setminus (O_1 \cup \cdots \cup O_{j-1})$. Then, two cases must be distinguished:
i) If $N_{j+1} = N_j \setminus O_j$ is null, then the hierarchy is: $O_1, \ldots, O_j$ and $O_{j+1} = N_{j+1}$.
ii) If N_{j+1} is not null, then we know, thanks to Theorem 5, that there exists a unique O_{j+1} predominant in N_{j+1}. This process can be iterated until the set of remaining states is a null set.

References

[1] K. J. Arrow. *Social Choice and Individual Values.* Cowles Foundations and Wiley - New-York, 1951.

[2] C. Boutilier. Toward a logic for qualitative decision theory. In *Proceedings of KR'94*, pages 75–86, 1994.

[3] D. Bouyssou. Some remarks on the notion of compensation in mcdm. *European Journal of Operational Research*, 26:150–160, 1986.

[4] D. Bouyssou, Th. Marchant, M. Pirlot, P. Perny, A. Tsoukiàs, and Ph. Vincke. *Evaluation and Decision Models: A critical perspective.* Kluwer, 2000.

[5] D. Bouyssou and M. Pirlot. *Non-transitive decomposable conjoint measurement* In: N. Meskens and M. Roubens (eds), Advances in Decision Analysis. Kluwer, 1999.

[6] D. Bouyssou and J. C. Vansnick. Noncompensatory and generalized noncompensatory preference structures. *Theory and Decision*, 21:251–266, 1986.

[7] R. Brafman and M. Tennenholtz. On the foundations of qualitative decision theory. In *Proceedings AAAI'96*, pages 1291–1296, 1996.

[8] R. Brafman and M. Tennenholtz. On the axiomatization of qualitative decision theory. In *Proceedings AAAI'97*, pages 76–81, 1997.

[9] R. Brafman and M. Tennenholtz. An axiomatic treatment of three qualitative decision criteria. *Journal of the ACM*, 47(3):452–482, 2000.

[10] D. Dubois, H. Fargier, and H. Prade. Decision-making under ordinal preferences and comparative uncertainty. In *Proceedings UAI'97 (157–164)*, 1997.

[11] D. Dubois and H. Prade. Conditional objects, possibility theory and default rules. In G. Crocco, L. Farinas del Cerro, and A. Herzig, editors, *Conditionals: From Philosophy to Computer Sciences*, pages 301–336. Oxford University Press, 1995.

[12] D. Dubois, H. Prade, and R. Sabbadin. *Qualitative decision theory with Sugeno Integrals* In: Proceedings of UAI'98, pages 121–128. 1998.

[13] Didier Duhois and Henri Prade. Numerical representations of acceptance. In *Proceedings UAI'95*, pages 149–156, 1995.

[14] H. Fargier and P. Perny. *Qualitative Decision Models under Uncertainty without the commensurability hypothesis* In: K.B. Laskey and H. Prade (eds), Proceedings of UAI'99, pages 188–195. 1999.

[15] P. C. Fishburn. Axioms for lexicographic preferences. *Review of Economic Studies*, 42:4.5–419, 1975.

[16] P. C. Fishburn. Noncompensatory preferences. *Synthèse*, 33:393–403, 1976.

[17] N. Friedman and J. Y. Halpern. Plausibility measures and default reasoning. In *Proceedings AAAI'96*, pages 1297–1304, 1996.

[18] M. Grabisch. The application of fuzzy integrals in multicriteria decision making. *European Journal of Operational Research*, 89:445–456, 1996.

[19] J. Y. Halpern. Defining relative likelihood in partially-ordered preferential structures. *Journal of A.I. Research*, 7:1–24, 1997.

[20] R. Keeney and H. Raiffa. *Decisions With Multiple Objectives: Preferences and Value Tradeoffs*. John Wiley and Sons, Inc., New York, 1976.

[21] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Art. Int.*, 44(1-2):167–207, 1990.

[22] H. Levy. Stochastic dominance and expected utility: survey and analysis. *Management Science*, 38(4):186–194, 1992.

[23] J.-L. Marichal. An axiomatic approach of the discrete choquet integral as a tool to aggregate interacting criteria. *IEEE Trans on Fuzzy Systems*, 8:800–807, 2000.

[24] M. Pirlot and Ph. Vincke. *Semiorders: properties, representations, applications*. Kluwer, 1997.

[25] B. Roy. How outranking relations helps multiple criteria decision making. In J. Cochrane and M. Zeleny, editors, *Multicriteria Decision Making*, pages 179–201. University of Carolina, 1973.

[26] B. Roy. *Multicriteria Methodology for Decision Aiding*. Kluwer, 1996.

[27] R. Sarin and P. Wakker. A simple axiomatization of non-additive expected utility. *Econometrica*, 60(6):1255–1272, 1991.

[28] L. J. Savage. *The Foundations of Statistics*. J. Wiley and Sons, New-York, 1954.

[29] D. Schmeidler. Integral representation without additivity. *Proceedings of the American Mathematical Society*, 97(2):255–261, 1986.

[30] A. K. Sen. *Social Choice Theory*. In: K.J. Arrow, M.D. Intrilligator (Eds), *Handbook of Mathematical Economics*, chapter 22, pages 1073–1181. Elsevier Sciences Publishers, North-Holland, 1986.

[31] M. Sugeno. *Theory of fuzzy integrals and its applications*. PhD thesis, Tokyo Institute of Technology, 1974.

[32] S. Tan and J. Pearl. Specification and evaluation of preferences under uncertainty. In Pietro Torasso Jon Doyle, Erik Sandewall, editor, *Proceedings KR'94*, pages 530–539, Bonn, FRG, May 1994. Morgan Kaufmann.

[33] R. Thomason. Desires and defaults: A framework for planning with inferred goals. In *Proceedings of KR'2000*, pages 702–713, 2000.

[34] Ph. Vincke. *Multicriteria Decision Aid*. Wiley, 1992.

[35] J. A. Weymark. Arrow's theorem with social quasi-orderings. *Public Choice*, 42(3):235–246, 1984.

Nonmonotonic Reasoning

A structural approach to default reasoning

Gabriele Kern-Isberner
Department of Computer Science
FernUniversität Hagen
D-58084 Hagen, Germany
gabriele.kern-isberner@fernuni-hagen.de

Abstract

In this paper, we present several approaches to default reasoning in a possibilistic framework which all are based on purely structural information, encoded by the *conditional structures* of worlds. Conditional structures symbolize the effects conditionals have on worlds and make conditional interactions transparent and computable, providing a sound algebraic framework for default inferences. Even the weakest of our inference relations, which merely demands a compatibility with conditional structures, proved to be quite well-behaved, according to widely accepted standards. The other two inference relations investigated here are based on faithful possibilistic realizations of conditional structures. Both these approaches allow to handle implicit priorities among the default rules under consideration and thereby combine structural with qualitative information.

1 Introduction

Default reasoning is based on using default rules as plausible, but uncertain rules for inference. It is only tentative, and its conclusions can be revised when new or more precise information becomes available. It is just this defeasibility, however, which makes default reasoning a major topic in Artificial Intelligence: Knowledge is seldom complete nor certain, and defeasible inference mechanisms provide a convenient framework to realize human reasoning. Within the last two decades, a great variety of different systems for default reasoning have been presented, in symbolic as well as in ordinal or numerical and even infinitesimal environments (see, e.g., [Adams, 1975; Reiter, 1980; Lehmann and Magidor, 1992; Goldszmidt *et al.*, 1993; Goldszmidt and Pearl, 1996; Benferhat *et al.*, 1992; 1999; 2000; Geffner, 1992; Lehmann, 1995]).

The crucial question in default reasoning is the following: Given a set $\mathcal{R}$ of default rules, which conclusions can be drawn from $\mathcal{R}$? Default reasoning is expected to be plausible, and it can even be bold sometimes, but it should never be arbitrary, so two further points have to be made clear: What are the mechanisms that give rise to these conclusions, and what are the properties the corresponding inference relation has? In [Benferhat *et al.*, 2000], the desirable properties a default inference relation $\mid\!\sim$ should have are summarized, as follows:

- *System P*: The inference relation should satisfy the following basic properties of system P (see, e.g., [Kraus *et al.*, 1990]):
  Reflexivity: $A \mid\!\sim A$ for all formulas A;
  Left Logical Equivalence:
  from $A \Leftrightarrow A' = \top$ and $A \mid\!\sim B$ deduce $A' \mid\!\sim B$;
  Right Weakening:
  from $B \models B'$ and $A \mid\!\sim B$ deduce $A \mid\!\sim B'$;
  OR: from $A \mid\!\sim C$ and $B \mid\!\sim C$ deduce $A \vee B \mid\!\sim C$;
  Cautious Monotony:
  from $A \mid\!\sim B$ and $A \mid\!\sim C$ deduce $A \wedge B \mid\!\sim C$;
  Cut: from $A \wedge B \mid\!\sim C$ and $A \mid\!\sim B$ deduce $A \mid\!\sim C$.
- *Rational Monotonicity*: If B can be (defeasibly) inferred from A, and $\neg C$ can *not* be inferred from A, then B can also be inferred from $A \wedge C$. Rational Monotonicity claims that information not in conflict with the knowledge given should not affect the possible conclusions.
- *Specificity*: More specific information should dominate general information. Knowing that Tweety is a bird, but also a member of the subclass "penguins", should yield the conclusion that Tweety cannot fly, in spite of the more general default rule that birds are normally able to fly.

- *Irrelevance*: Atoms not occurring in any of the defaults should be irrelevant to default conclusions.
- *Property inheritance*: Members of a subclass which are exceptional with respect to one property should be able to inherit other properties from the superclass. In our penguin example, we would expect Tweety to lay eggs, as all birds do, although he is an exceptional bird.
- *Ambiguity preservation*: Conflicts between defaults should not be resolved arbitrarily. In case that we have two equally strong arguments, one in favor of and one against a proposition, no conclusion should be drawn at all.
- *Duplication stability*[1]: Duplications of defaults in the rule base should not affect the conclusions.

In this paper, we will present methods for reasoning with defaults which are based on mainly structural arguments and which satisfy most of the properties listed above, with the only exception of *Rational Monotonicity*. The structures that are crucial for the inference process here are *conditional structures*, assigned to possible worlds and reflecting a world's attitude with respect to the set of default rules under consideration. The first inference relation, $\mid\!\sim^{s}_{\mathcal{R}}$, is actually quite a weak one and only needs a purely qualitative environment. Hence, it is based on comparative possibility distributions [Benferhat *et al.*, 1999] which are compatible with conditional structures. The second inference relation, $\mid\!\sim^{c}_{\mathcal{R}}$, is a more sophisticated one; it is realized by considering *c-representations*, i.e. possibility distributions which are built from conditional structures in a straightforward way. $\mid\!\sim^{c}_{\mathcal{R}}$ is also based on structural information, but, in using numbers, it allows of taking implicit information about priorities between defaults into account. Therefore, it is stronger than $\mid\!\sim^{s}_{\mathcal{R}}$ and provides a more convenient environment for inferences. The last inference relation, $\mid\!\sim^{lcc}_{\mathcal{R}}$, is obtained by focusing on least specific and least committed c-representations. It turns out to be a possibilistic realization of $\mid\!\sim_{lcd}$, the default inference relation of [Benferhat *et al.*, 2000] using lcd-belief functions.

This paper is organized as follows: The following section recalls the notion of (comparative) possibility distributions and how these can be used to validate default rules (or *conditionals*). Section 3 gives a brief overview on the theory of conditional structures which is basic to the methods of this paper. In particular, we define a partial ordering on conditional structures of possible worlds here. In Section 4, we introduce the notion of *structural compatibility* of comparative possibility distributions with sets of defaults, and investigate the corresponding structural inference relation. Section 5 deals with *c-representations* of default rules which can be regarded as numerical realizations of algebraic conditional structures. This section also introduces *lcc-representations* as a special type of c-representations. Section 6 concludes this paper by highlighting its main results, and by pointing out connections to other approaches to uncertain reasoning.

[1]This property is referred to as *syntax independence* or *redundancy* in [Benferhat *et al.*, 2000]. We prefer *duplication stability* here since the notions used in [Benferhat *et al.*, 2000] suggest quite stronger properties to hold.

2 Default rules in possibility theory

Default rules describe what is normally the case, or what is expected. In this paper, we will take quite a general view and understand default rules simply as plausible relationships "*If A then B*" between two formulas A, B of a (finitely generated) propositional language $\mathcal{L}$. We will use here the conditional notation $(B|A)$ for a default rule, with *antecedent* or *premise* A, and *consequent* B, and also speak of default rules simply as "conditionals". Let $(\mathcal{L} \mid \mathcal{L})$ denote the set of all default rules $(B|A)$ with $A, B \in \mathcal{L}$. We will only consider finite sets $\mathcal{R} \subseteq (\mathcal{L} \mid \mathcal{L})$ of conditionals. To simplify notation, we will write $\overline{A}$ instead of $\neg A$, and AB instead of $A \wedge B$, for $A, B \in \mathcal{L}$. Let Ω be the complete set of interpretations of $\mathcal{L}$, where each $\omega \in \Omega$ is taken to be a *possible world* for $\mathcal{L}$.

A *possibility distribution* is a mapping $\pi : \Omega \to [0,1]$, where each $\pi(\omega)$ reflects the degree of possibility an intelligent agent assigns to ω. π induces a *possibility measure* $\Pi : \mathcal{L} \to [0,1], \Pi(A) = \max\{\pi(\omega) \mid \omega \models A\}$ and a dual *necessity measure* $N = N_\pi : \mathcal{L} \to [0,1], N(A) = 1 - \Pi(\overline{A})$ (cf., e.g., [Dubois *et al.*, 1994]). We will not distinguish between a possibility distribution, π, on Ω, and its induced possibility measure, Π, on $\mathcal{L}$, and will use the same symbol π for both.

A default rule $(B|A)$ *is represented* (or *accepted*) *by* π, $\pi \models (B|A)$, iff $\pi(AB) > \pi(A\overline{B})$. So, in accordance with intuition, a default rule $(B|A)$ is accepted in the epistemic state modeled by a possibility distribution, if its confirmation (AB) is considered to be more possible (or plausible) than its refutation $(A\overline{B})$. We assume all possibility distributions to be normalized, i.e. $\pi(\top) = 1$, and, only for conciseness of presentation, to be positive, i.e. $\pi(\omega) > 0$ for all $\omega \in \Omega$.

Each possibility distribution induces uniquely a complete pre-ordering on possible worlds by $\omega_1 \succeq_\pi \omega_2$ iff $\pi(\omega_1) \geqslant \pi(\omega_2)$. Different possibility distributions,

however, may give rise to the same pre-ordering $\succeq_\pi$. If one is only concerned with qualitative relationships, one can abstract from numbers and focus on such complete pre-orderings $\succeq_\pi$ on Ω, which are called *comparative possibility distributions* (cf. also [Benferhat *et al.*, 1999]). The subscript π does not refer to a particular possibility distribution, but is to symbolize possibility or plausibility, in general. In the usual way, we define $\omega \succ_\pi \omega'$ by $\omega \succeq_\pi \omega'$ and not $(\omega' \succeq_\pi \omega)$, and $\omega \approx_\pi \omega'$ iff both $\omega \succeq_\pi \omega'$ and $\omega' \succeq_\pi \omega$.

From a comparative possibility distribution $\succeq_\pi$, a *comparative possibility measure* on $\mathcal{L}$, also denoted by $\succeq_\pi$, can be obtained by $A \succeq_\pi B$ ($A \succ_\pi B$) iff there exists $\omega \models A$ such that for all $\omega' \models B$, $\omega \succeq_\pi \omega'$ ($\omega \succ_\pi \omega'$). $\succeq_\pi$ defines a complete pre-ordering on the set of formulas $\mathcal{L}$, in particular, $A \succ_\pi \bot$ for any consistent formula $A \in \mathcal{L}$. Again we hope that no confusion may arise when using the same symbol $\succeq_\pi$ for comparative possibility distributions and comparative possibility measures.

Like possibility distributions, each comparative possibility distribution provides an epistemic structure to validate default rules: A comparative possibility distribution $\succ_\pi$ *accepts* (or *represents*) *a default rule* $(B|A)$ iff $AB \succ_\pi A\overline{B}$.

For (comparative) possibility distributions, the definition of representing default rules is easily extended to sets $\mathcal{R}$ of default rules by postulating that each rule in $\mathcal{R}$ must be represented. In particular, the set of all comparative possibility distributions which accept $\mathcal{R} = \{(B_1|A_1), \ldots, (B_n|A_n)\}$ is denoted by $\Pi^{\succeq}(\mathcal{R})$:

$$\Pi^{\succeq}(\mathcal{R}) = \{\succeq_\pi |\ A_iB_i \succ_\pi A_i\overline{B}_i \text{ for all } i, 1 \leqslant i \leqslant n\}$$

3 Conditional structures

The theory of conditional structures has been developed in detail in several publications (cf. [Kern-Isberner, 2000; 2001a; 2001c; 2001b]); so we only summarize the main ideas here.

From the theory of conditionals, the following distinction concerning the possible relationships between worlds and conditionals is well-known: A world $\omega \in \Omega$ is said to *verify* a conditional (default rule) $(B|A)$, if $\omega \models AB$; it *falsifies* $(B|A)$, if $\omega \models A\overline{B}$; if $\omega \models \overline{A}$, then $(B|A)$ is not applicable to ω (cf. e.g. [DeFinetti, 1974; Calabrese, 1991]).

Intuitively, representing a conditional as a plausible conclusion in an epistemic state means to make – at least some – worlds verifying the conditional more plausible than the worlds falsifying it. In this sense, conditionals have effects on possible worlds – more exactly, on their degrees of plausibility or possibility, shifting them appropriately to establish the intended relationship. When we consider (finite) sets of conditionals $\mathcal{R} = \{(B_1|A_1), \ldots, (B_n|A_n)\} \subseteq (\mathcal{L} \mid \mathcal{L})$, we have to identify the effect of each conditional in $\mathcal{R}$ on worlds in Ω. To this end, we use a pair of abstract symbols, $\mathrm{a}_i^+, \mathrm{a}_i^-$, that we associate to each conditional $(B_i|A_i)$ in $\mathcal{R}$. Moreover, we will make use of a group structure to represent the joint impact of conditionals on worlds as products.

So let $\mathcal{F}_\mathcal{R} = \langle \mathrm{a}_1^+, \mathrm{a}_1^-, \ldots, \mathrm{a}_n^+, \mathrm{a}_n^- \rangle$ be the free abelian group with generators $\mathrm{a}_1^+, \mathrm{a}_1^-, \ldots, \mathrm{a}_n^+, \mathrm{a}_n^-$, i.e. $\mathcal{F}_\mathcal{R}$ consists of all elements of the form $(\mathrm{a}_1^+)^{r_1}(\mathrm{a}_1^-)^{s_1} \ldots (\mathrm{a}_n^+)^{r_n}(\mathrm{a}_n^-)^{s_n}$ with integers $r_i, s_i \in \mathbb{Z}$ (the ring of integers) (cf. e.g. [Lyndon and Schupp, 1977; Fine and Rosenberger, 1999]). Note that, although we will speak of *multiplication* and *products* in $\mathcal{F}_\mathcal{R}$, the generators of $\mathcal{F}_\mathcal{R}$ are merely juxtaposed, like words. Moreover, it should be clear that it does not matter which symbols $\mathrm{a}_i^+, \mathrm{a}_i^-$ have actually been chosen – taking a set of different symbols will yield isomorphic structures.

For each $i, 1 \leqslant i \leqslant n$, we define a function $\sigma_i = \sigma_{(B_i|A_i)} : \Omega \to \mathcal{F}_\mathcal{R}$ by setting

$$\sigma_i(\omega) := \sigma_{(B_i|A_i)}(\omega) := \begin{cases} \mathrm{a}_i^+ & \text{if } \omega \models A_iB_i \\ \mathrm{a}_i^- & \text{if } \omega \models A_i\overline{B_i} \\ 1 & \text{if } \omega \models \overline{A_i} \end{cases}$$

$\sigma_i(\omega)$ represents the manner in which the conditional $(B_i|A_i)$ applies to the possible world ω. The neutral element 1 of $\mathcal{F}_\mathcal{R}$ represents the non-applicability of $(B_i|A_i)$ in case that the antecedent A_i is not satisfied, so the neutral group element corresponds to a neutral attitude with respect to the conditional. The function $\sigma_\mathcal{R} : \Omega \to \mathcal{F}_\mathcal{R}$,

$$\sigma_\mathcal{R}(\omega) := \prod_{1 \leqslant i \leqslant n} \sigma_i(\omega) = \prod_{\substack{1 \leqslant i \leqslant n \\ \omega \models A_iB_i}} \mathrm{a}_i^+ \prod_{\substack{1 \leqslant i \leqslant n \\ \omega \models A_i\overline{B}_i}} \mathrm{a}_i^-$$

describes the all-over effect of $\mathcal{R}$ on ω. $\sigma_\mathcal{R}(\omega)$ is called the *conditional structure of* ω *with respect to* $\mathcal{R}$. Since $\mathcal{F}_\mathcal{R}$ is a free (abelian) group, the conditional structures of worlds are uniquely determined by its σ_i-components: For any two worlds ω_1, ω_2, we have

$$\sigma_\mathcal{R}(\omega_1) = \sigma_\mathcal{R}(\omega_2) \quad \text{iff} \quad \sigma_i(\omega_1) = \sigma_i(\omega_2) \text{ for all } 1 \leqslant i \leqslant n \qquad (1)$$

So, each conditional is taken to be an independent piece of information. To make notations more convenient in the following, we write $\sigma_\mathcal{R}(A) := \{\sigma_\mathcal{R}(\omega) \mid \omega \in \Omega, \omega \models A\}$ for the set of conditional structures appertaining to the models of $A \in \mathcal{L}$.

We will illustrate this notion of conditional structures in the following example which extends the well-known *Nixon diamond*:

Example 1 Let $\mathcal{R}$ consist of the following conditionals:

$r_1 : (p|q)$ *Quakers are pacifists.*
$r_2 : (\overline{p}|r)$ *Republicans are not pacifists.*
$r_3 : (a|q)$ *Quakers are Americans.*
$r_4 : (b|a)$ *Americans like baseball.*
$r_5 : (\overline{b}|q)$ *Quakers do not like baseball.*

The conditional structure of a possible world, say $pqrab$, is calculated in the following way: $pqrab$ verifies the first conditional r_1 ($pqrab \models pq$), so we have $\sigma_1(pqrab) = \mathrm{a}_1^+$. $pqrab$, however, falsifies the second conditional $r_2 = (\overline{p}|r)$, thus $\sigma_2(pqrab) = \mathrm{a}_2^-$. In the same way, $\sigma_3(pqrab) = \mathrm{a}_3^+, \sigma_4(pqrab) = \mathrm{a}_4^+, \sigma_5(pqrab) = \mathrm{a}_5^-$. We obtain $\sigma_{\mathcal{R}}(pqrab) = \mathrm{a}_1^+\mathrm{a}_2^-\mathrm{a}_3^+\mathrm{a}_4^+\mathrm{a}_5^-$. ∎

The conditional structures of worlds are very special elements of $\mathcal{F}_{\mathcal{R}}$: For each i, $1 \leqslant i \leqslant n$, at most one of $\mathrm{a}_i^+, \mathrm{a}_i^-$ occurs once in $\sigma_{\mathcal{R}}(\omega)$ since each conditional in $\mathcal{R}$ applies to ω in a well-defined way. Group elements of this kind will be called simple products:

Definition 2 An element $\mathrm{a} \in \mathcal{F}_{\mathcal{R}}$ is called a *simple product* iff for each i, $1 \leqslant i \leqslant n$, at most one of $\mathrm{a}_i^+, \mathrm{a}_i^-$ occurs in a, with exponent 1.

The set of all simple products in $\mathcal{F}_{\mathcal{R}}$ will be denoted by $\mathcal{SP}_{\mathcal{R}}$.

So every conditional structure $\sigma_{\mathcal{R}}(\omega)$ of a possible world $\omega \in \Omega$ is a simple product, i.e. $\sigma_{\mathcal{R}}(\Omega) \subseteq \mathcal{SP}_{\mathcal{R}}$. In general, however, we will have $\sigma_{\mathcal{R}}(\Omega) \neq \mathcal{SP}_{\mathcal{R}}$.

Next, we define a partial ordering $>$ on simple products. To this end, we have to check which of the generators of $\mathcal{F}_{\mathcal{R}}$ actually occur in a simple product. This is done by the aid of *signature functions* $sign_i : \mathcal{SP}_{\mathcal{R}} \to \{+, -, 0\}$, which are defined for each conditional $(B_i|A_i) \in \mathcal{R}$, $1 \leqslant i \leqslant n$, by

$$sign_i(\mathrm{a}) := \begin{cases} +, & \text{if } \mathrm{a}_i^+ \text{ occurs in a} \\ -, & \text{if } \mathrm{a}_i^- \text{ occurs in a} \\ 0, & \text{otherwise} \end{cases}$$

Note that the signature functions $sign_i$ (defined on group elements) correspond to the structure functions σ_i (defined on possible worlds) in that $sign_i(\sigma_{\mathcal{R}}(\omega))$ reflects the behavior of ω with respect to $(B_i|A_i)$ independent of the symbols $\mathrm{a}_i^+, \mathrm{a}_i^-$. In particular, from (1), we have that $\sigma_{\mathcal{R}}(\omega_1) = \sigma_{\mathcal{R}}(\omega_2)$ iff $sign_i(\sigma_{\mathcal{R}}(\omega_1)) = sign_i(\sigma_{\mathcal{R}}(\omega_2))$ for all $1 \leqslant i \leqslant n$, for any two worlds $\omega_1, \omega_2 \in \Omega$.

Let $\mathrm{a}, \mathrm{b} \in \mathcal{SP}_{\mathcal{R}}$ be simple products. We define $\mathrm{a} > \mathrm{b}$ iff for each i, $1 \leqslant i \leqslant n$,

- $sign_i(\mathrm{a}) \neq 0$ iff $sign_i(\mathrm{b}) \neq 0$, and
- $sign_i(\mathrm{a}) = -$ implies $sign_i(\mathrm{b}) = -$, and
- there is at least one i, $1 \leqslant i \leqslant n$, such that $sign_i(\mathrm{a}) = +$ and $sign_i(\mathrm{b}) = -$.

So, for instance, $\mathrm{a}_1^+\mathrm{a}_2^+\mathrm{a}_3^+ > \mathrm{a}_1^+\mathrm{a}_2^+\mathrm{a}_3^- > \mathrm{a}_1^+\mathrm{a}_2^-\mathrm{a}_3^- > \mathrm{a}_1^-\mathrm{a}_2^-\mathrm{a}_3^-$, whereas $\mathrm{a}_1^+\mathrm{a}_2^-\mathrm{a}_3^+$ and $\mathrm{a}_1^+\mathrm{a}_2^+\mathrm{a}_3^-$ are not comparable with respect to $>$.

This partial ordering $>$ can now be used to compare the conditional structures of possible worlds $\omega_1, \omega_2 \in \Omega$: We have $\sigma_{\mathcal{R}}(\omega_1) > \sigma_{\mathcal{R}}(\omega_2)$ iff the same conditionals in $\mathcal{R}$ are applicable to either of them, and each conditional, that is falsified by ω_1, is also falsified by ω_2, but there is at least one conditional in $\mathcal{R}$ which is verified by ω_1 and falsified by ω_2. Note that only the behavior of worlds with respect to the same conditionals is relevant to compare their conditional structures. $>$ provides no means to relate the conditional structures, for instance, of two worlds ω_1, ω_2 such that ω_1 verifies $(B_1|A_1)$ but falsifies $(B_2|A_2)$, say, and ω_2 verifies $(B_2|A_2)$ but falsifies $(B_1|A_1)$. No information about default priorities and the like is taken into account for the purely structural partial ordering $>$, as is e.g. done in the *prioritized preferential structures* of [Geffner and Pearl, 1992]. Note also that we do not focus here merely on which conditionals are falsified by the corresponding worlds, which is a most common approach in default reasoning to make conditionals fit a two-valued logic (see, e.g., [Geffner and Pearl, 1992; Goldszmidt and Pearl, 1996; Benferhat *et al.*, 2000]). We also distinguish between the case that a conditional is verified by a world, and the case that it is not applicable to that world, thus respecting the inherent three-valued nature of a conditional.

4 Default reasoning compatible with conditional structures

We will now consider comparative possibility distributions that not only represent a set of conditionals $\mathcal{R}$, but which also observe the information carried by the conditional structure of worlds. Such a comparative possibility distribution should not distinguish between worlds having the same conditional structures, and should give priority to those worlds that verify conditionals, in order to establish conditional beliefs:

Definition 3 A comparative possibility distribution $\succeq_\pi$ is said to be *structurally compatible* (*s-compatible*) *with* $\mathcal{R} \subseteq (\mathcal{L} \mid \mathcal{L})$ iff it represents $\mathcal{R}$ (i.e. $\succeq_\pi \in \Pi^{\succeq}(\mathcal{R})$) and for all possible worlds $\omega, \omega' \in \Omega$, the following conditions are fulfilled:

- $\sigma_{\mathcal{R}}(\omega) = \sigma_{\mathcal{R}}(\omega')$ implies $\omega \approx_\pi \omega'$;
- $\sigma_{\mathcal{R}}(\omega) > \sigma_{\mathcal{R}}(\omega')$ implies $\omega \succ_\pi \omega'$.

The set of comparative possibility distributions which are s-compatible with $\mathcal{R}$ is denoted by $\Pi^s(\mathcal{R})$. Considering all comparative possibility distributions which are s-compatible with $\mathcal{R}$ gives rise to a (nonmonotonic) *structural inference relation* $\mid\!\sim^s_{\mathcal{R}}$ on $\mathcal{L}$:

$$A \mid\!\sim^s_{\mathcal{R}} B \quad \text{iff} \quad AB \succ_\pi A\overline{B} \text{ for all } \succeq_\pi \in \Pi^s(\mathcal{R})$$

The following proposition shows how structural default reasoning can be done by considering conditional structures:

Proposition 4 *Let A, B be two propositional formulas such that for all $\omega \models A\overline{B}$, there is $\omega' \models AB$ such that $\sigma_{\mathcal{R}}(\omega') > \sigma_{\mathcal{R}}(\omega)$. Then $A \mid\!\sim^s_{\mathcal{R}} B$.*

Proof. We have to show that $AB \succ_\pi A\overline{B}$ for all s-compatible comparative possibility distributions $\succeq_\pi \in \Pi^s(\mathcal{R})$. So let $\succeq_\pi \in \Pi^s(\mathcal{R})$, and let $\omega_1 \models AB$ be maximal with respect to $\succeq_\pi$. Let $\omega \models A\overline{B}$. By presupposition, there is $\omega' \models AB$ with $\sigma_{\mathcal{R}}(\omega') > \sigma_{\mathcal{R}}(\omega)$. Due to s-compatibility, this implies $\omega' \succ_\pi \omega$, hence $\omega_1 \succeq_\pi \omega' \succ_\pi \omega$ for all $\omega \models A\overline{B}$. This proves $AB \succ_\pi A\overline{B}$. □

We will now check which of the properties of our catalogue (see Section 1) hold for $\mid\!\sim^s_{\mathcal{R}}$. It is straightforward to verify all properties of system P, so $\mid\!\sim^s_{\mathcal{R}}$ satisfies the very basic properties a nonmonotonic inference operation should obey. Furthermore, $\mid\!\sim^s_{\mathcal{R}}$ also respects *specificity*, as the simple penguin-example shows:

Example 5 Let $\mathcal{R}$ consist of the conditionals $(f|b)$ (*birds fly*), $(b|p)$ (*penguins are birds*), and $(\overline{f}|p)$ (*penguins do not fly*), with symbols $\mathbf{a}_1^+, \mathbf{a}_1^-, \mathbf{a}_2^+, \mathbf{a}_2^-, \mathbf{a}_3^+, \mathbf{a}_3^-$ associated to the conditionals (in that order). The conditional structures are listed in the following table:

ω	$\sigma_{\mathcal{R}}(\omega)$	ω	$\sigma_{\mathcal{R}}(\omega)$
pbf	$\mathbf{a}_1^+\mathbf{a}_2^+\mathbf{a}_3^-$	$\overline{p}bf$	$\mathbf{a}_1^+$
$pb\overline{f}$	$\mathbf{a}_1^-\mathbf{a}_2^+\mathbf{a}_3^+$	$\overline{p}b\overline{f}$	$\mathbf{a}_1^-$
$p\overline{b}f$	$\mathbf{a}_2^-\mathbf{a}_3^-$	$\overline{p}\overline{b}f$	1
$p\overline{b}\,\overline{f}$	$\mathbf{a}_2^-\mathbf{a}_3^+$	$\overline{p}\overline{b}\,\overline{f}$	1

Specificity would postulate here that from knowing that an individual is both a penguin and a bird, we could derive that this individual can not fly – the more specific information is expected to override the general knowledge.

Let $\succeq_\pi \in \Pi^s(\mathcal{R})$. Then $\succeq_\pi$ represents $\mathcal{R}$, so, in particular, we have $pb \succ_\pi p\overline{b}$ and $p\overline{f} \succ_\pi pf$, which means $\max\{pbf, pb\overline{f}\} \succ_\pi \max\{p\overline{b}f, p\overline{b}\,\overline{f}\}$ and $\max\{pb\overline{f}, p\overline{b}\,\overline{f}\} \succ_\pi \max\{pbf, p\overline{b}f\}$. Furthermore, s-compatibility enforces $p\overline{b}\,\overline{f} \succ_\pi p\overline{b}f$ and $\overline{p}bf \succ_\pi \overline{p}b\overline{f}$. The first inequality, together with $pb \succ_\pi p\overline{b}$, implies $\max\{pbf, pb\overline{f}\} \succ_\pi p\overline{b}\,\overline{f}$. Now, if $pbf \succeq_\pi pb\overline{f}$, then $pbf \succ_\pi p\overline{b}\,\overline{f} \succ_\pi pbf$ would follow, a contradiction. Hence $pb\overline{f} \succ_\pi pbf$ for all $\succeq_\pi \in \Pi^s(\mathcal{R})$. Therefore, we have $pb \mid\!\sim^s_{\mathcal{R}} \overline{f}$. ■

The next proposition shows that $\mid\!\sim^s_{\mathcal{R}}$ also fulfills *irrelevance*:

Proposition 6 *Let $A, B, C \in \mathcal{L}$ be propositions such that C is composed of atomic propositions not occurring in any of the conditionals in $\mathcal{R}$, nor in $(B|A)$. Then $A \mid\!\sim^s_{\mathcal{R}} B$ iff $AC \mid\!\sim^s_{\mathcal{R}} BC$.*

Proof. Let $\mathcal{V}_1 \subset \mathcal{V}$ be the set of atomic propositions occurring in C, let $\mathcal{V}_2 := \mathcal{V} - \mathcal{V}_1$, $\mathcal{V}_1, \mathcal{V}_2 \neq \emptyset$. Then each possible world $\omega \in \Omega$ can be written as $\omega = \nu_1\nu_2$ with ν_1, ν_2 possible worlds (i.e. interpretations) over $\mathcal{V}_1, \mathcal{V}_2$, respectively. The conditional structure $\sigma_{\mathcal{R}}(\omega)$ is determined by the $\mathcal{V}_2$-part of ω, since no variable in $\mathcal{V}_1$ occurs in $\mathcal{R}$. Therefore, $\sigma_{\mathcal{R}}(\nu_1\nu_2) = \sigma_{\mathcal{R}}(\nu_1'\nu_2)$ for all ν_1, ν_1', and for a fixed (but arbitrary) ν_2.

Let $\succ_\pi \in \Pi^s(\mathcal{R})$, and let $\dot{B}$ be one of $B, \overline{B}$. In particular, for any $\succ_\pi$-maximal world $\omega = \nu_1\nu_2$ with $\omega \models A\dot{B}C$, ν_1' can be chosen such that $\nu_1'\nu_2 \models A\dot{B}\overline{C}$ and $\sigma_{\mathcal{R}}(\nu_1\nu_2) = \sigma_{\mathcal{R}}(\nu_1'\nu_2)$. Hence $\nu_1\nu_2 \approx_\pi \nu_1'\nu_2$. A similar argumentation for the $\succ_\pi$-maximal worlds of $A\dot{B}\overline{C}$ shows $A\dot{B}C \approx_\pi A\dot{B}\overline{C} \approx_\pi A\dot{B}$. Therefore, $AB \succ_\pi A\overline{B}$ iff $ABC \succ_\pi A\overline{B}C$ for each $\succ_\pi \in \Pi^s(\mathcal{R})$. □

The next two examples will illustrate that also exceptional subtypes can inherit properties from their superclass:

Example 7 Let $\mathcal{R}$ consist of the following conditionals:

r_1: $(f|b)$ *birds fly*
r_2: $(\overline{f}|p)$ *penguins do not fly*
r_3: $(b|p)$ *penguins are birds*
r_4: $(l|b)$ *birds have legs*

Let $\mathbf{a}_i^+, \mathbf{a}_i^-$ be associated with conditional r_i, $1 \leqslant i \leqslant 4$. Then $pb \mid\!\sim^s_{\mathcal{R}} l$ can be derived, i.e. penguin-birds are supposed to have legs: Due to Proposition 4, we only have to check the conditional structures in $\sigma_{\mathcal{R}}(pbl) = \{\mathbf{a}_1^+\mathbf{a}_2^-\mathbf{a}_3^+\mathbf{a}_4^+, \mathbf{a}_1^-\mathbf{a}_2^+\mathbf{a}_3^+\mathbf{a}_4^+\}$ and $\sigma_{\mathcal{R}}(pb\overline{l}) = \{\mathbf{a}_1^+\mathbf{a}_2^-\mathbf{a}_3^+\mathbf{a}_4^-, \mathbf{a}_1^-\mathbf{a}_2^+\mathbf{a}_3^+\mathbf{a}_4^-\}$. Since $\mathbf{a}_1^+\mathbf{a}_2^-\mathbf{a}_3^+\mathbf{a}_4^+ > \mathbf{a}_1^+\mathbf{a}_2^-\mathbf{a}_3^+\mathbf{a}_4^-$ and $\mathbf{a}_1^-\mathbf{a}_2^+\mathbf{a}_3^+\mathbf{a}_4^+ > \mathbf{a}_1^-\mathbf{a}_2^+\mathbf{a}_3^+\mathbf{a}_4^-$, we have $pb \mid\!\sim^s_{\mathcal{R}} l$. ■

Example 8 Here, the rule base $\mathcal{R}$ represents knowledge about *Swedes* having the properties *blond* and *tall*: $\mathcal{R} = \{(b|s), (t|s)\}$ (*Swedes are blond* and *Swedes are tall*). Then, short Swedes are also expected to be blond: Let $\mathrm{a}_1^+, \mathrm{a}_1^-$ be associated with the first conditional, and $\mathrm{a}_2^+, \mathrm{a}_2^-$ with the second one. Since $\sigma_{\mathcal{R}}(sb\bar{t}) = \mathrm{a}_1^+\mathrm{a}_2^- > \mathrm{a}_1^-\mathrm{a}_2^- = \sigma_{\mathcal{R}}(s\bar{b}\,\bar{t})$, Proposition 4 implies $s\bar{t} \mid\!\sim^s_{\mathcal{R}} b$. ■

In continuing Example 1, we show that $\mid\!\sim^s_{\mathcal{R}}$ preserves ambiguity.

Example 1 [(continued)] $\mid\!\sim^s_{\mathcal{R}}$ yields no conclusion as to whether a person who is a quaker and a Republican is a pacifist or not. This is not surprising, since $\mid\!\sim^s_{\mathcal{R}}$ is actually quite weak. It is noticeable, however, *how* ambiguity is preserved by $\mid\!\sim^s_{\mathcal{R}}$ by considering the conditional structures of pqr and $\bar{p}qr$ which are listed below:

ω	$\sigma_{\mathcal{R}}(\omega)$	ω	$\sigma_{\mathcal{R}}(\omega)$
$pqrab$	$\mathrm{a}_1^+\mathrm{a}_2^-\mathrm{a}_3^+\mathrm{a}_4^+\mathrm{a}_5^-$	$\bar{p}qrab$	$\mathrm{a}_1^-\mathrm{a}_2^+\mathrm{a}_3^+\mathrm{a}_4^+\mathrm{a}_5^-$
$pqra\bar{b}$	$\mathrm{a}_1^+\mathrm{a}_2^-\mathrm{a}_3^+\mathrm{a}_4^-\mathrm{a}_5^+$	$\bar{p}qra\bar{b}$	$\mathrm{a}_1^-\mathrm{a}_2^+\mathrm{a}_3^+\mathrm{a}_4^-\mathrm{a}_5^+$
$pqr\bar{a}b$	$\mathrm{a}_1^+\mathrm{a}_2^-\mathrm{a}_3^-\mathrm{a}_5^-$	$\bar{p}qr\bar{a}b$	$\mathrm{a}_1^-\mathrm{a}_2^+\mathrm{a}_3^-\mathrm{a}_5^-$
$pqr\bar{a}\bar{b}$	$\mathrm{a}_1^+\mathrm{a}_2^-\mathrm{a}_3^-\mathrm{a}_5^+$	$\bar{p}qr\bar{a}\bar{b}$	$\mathrm{a}_1^-\mathrm{a}_2^+\mathrm{a}_3^-\mathrm{a}_5^+$

As to the third, fourth and fifth conditional in $\mathcal{R}$, there is a complete symmetry in the conditional structures of the worlds satisfying pqr and $\bar{p}qr$, respectively. More exactly, we have $\sigma_{\mathcal{R}}(pqr\dot{a}\dot{b}) = \sigma_{\mathcal{R}}(\bar{p}qr\dot{a}\dot{b})$ for $\dot{a} \in \{a, \bar{a}\}, \dot{b} \in \{b, \bar{b}\}$, and it is clearly seen that the ambiguity is only due to the conflict between the first and the second conditional. The other conditionals in $\mathcal{R}$ do not have any influence on the decision whether a person who is both a quaker and a Republican is a pacifist or not, as one should expect. Note that standard possibilistic inference does not preserve ambiguity in this example (cf. [Dubois and Prade, 1996]). ■

Finally, the following proposition proves that $\mid\!\sim^s_{\mathcal{R}}$ is not altered by repeating rules in the knowledge base:

Proposition 9 *Let* $\mathcal{R} = \{(B_1|A_1), \ldots, (B_n|A_n)\}$ *be a finite set of conditionals, and let* $\mathcal{R}' = \{(B_0|A_0), (B_1|A_1), \ldots, (B_n|A_n)\}$ *contain the same conditionals* $(B_1|A_1), \ldots, (B_n|A_n)$ *with a duplicated first conditional* $(B_0|A_0) \equiv (B_1|A_1)$. *Then* $A \mid\!\sim^s_{\mathcal{R}} B$ *iff* $A \mid\!\sim^s_{\mathcal{R}'} B$ *for* $A, B \in \mathcal{L}$.

Proof. Let $\sigma_{\mathcal{R}}, \sigma_{\mathcal{R}'}$ be two conditional structure homomorphisms appertaining to $\mathcal{R}, \mathcal{R}'$, respectively, with signature functions $sign_i, sign'_j$, $1 \leqslant i \leqslant n$, $0 \leqslant j \leqslant n$. Then for all $\omega \in \Omega$ and for all i, $2 \leqslant i \leqslant n$, we have $sign_i(\sigma_{\mathcal{R}}(\omega)) = sign'_i(\sigma_{\mathcal{R}'}(\omega))$, and $sign_1(\sigma_{\mathcal{R}}(\omega)) = sign'_1(\sigma_{\mathcal{R}'}(\omega)) = sign'_0(\sigma_{\mathcal{R}'}(\omega))$. Therefore $\sigma_{\mathcal{R}}(\omega_1) = \sigma_{\mathcal{R}}(\omega_2)$ iff $\sigma_{\mathcal{R}'}(\omega_1) = \sigma_{\mathcal{R}'}(\omega_2)$, and $\sigma_{\mathcal{R}}(\omega_1) > \sigma_{\mathcal{R}}(\omega_2)$ iff $\sigma_{\mathcal{R}'}(\omega_1) > \sigma_{\mathcal{R}'}(\omega_2)$, for $\omega_1, \omega_2 \in \Omega$. Hence, $\Pi^s(\mathcal{R}) = \Pi^s(\mathcal{R}')$, and the statement follows trivially. □

So, nearly all of the postulates listed in Section 1 could be shown to be fulfilled by the structural inference relation $\mid\!\sim^s_{\mathcal{R}}$. Only Rational Monotonicity has not yet been considered, and here we observe a failure – $\mid\!\sim^s_{\mathcal{R}}$ does not satisfy Rational Monotonicity, as the following example shows.

Example 10 Let a, b, c be atoms of the language $\mathcal{L}$, and let $\mathcal{R} = \{(a\bar{b}c|ab\bar{c} \vee a\bar{b}c), (abc|abc \vee a\bar{b}\bar{c})\}$, with α_1^+, α_1^- be associated with the first conditional, and α_2^+, α_2^- with the second. Then $a \mid\!\sim^s_{\mathcal{R}} c$, since $\sigma_{\mathcal{R}}(ac) = \{\alpha_1^+, \alpha_2^+\}$ and $\sigma_{\mathcal{R}}(a\bar{c}) = \{\alpha_1^-, \alpha_2^-\}$. On the other hand, $\sigma_{\mathcal{R}}(ab) = \{\alpha_1^-, \alpha_2^+\}$, and $\sigma_{\mathcal{R}}(a\bar{b}) = \{\alpha_1^+, \alpha_2^-\}$, so we can not derive that $a \mid\!\sim^s_{\mathcal{R}} b$. But also $\alpha_2^+ = \sigma_{\mathcal{R}}(abc)$ and $\alpha_1^- = \sigma_{\mathcal{R}}(ab\bar{c})$ are incomparable with respect to $>$, so we do not have $ab \mid\!\sim^s_{\mathcal{R}} c$. ■

This example is quite a formal one and gives little evidence whether this inference behavior is more reasonable than Rational Monotonicity, or the other way around. There are, however, other works that question Rational Monotonicity as a general inference rule (for a more intuitive counterexample, cf. [Benferhat *et al.*, 2000, p. 35]). The point with Rational Monotonicity is, that it considers only bilateral relationships (between A and B, respectively A and C), but is not apt to take complex interdependencies between all three formulas A, B, C, or between the corresponding conditionals, respectively, into account. A way to make interactions between conditionals explicit is to use the two relations $\sqsubseteq$ (*subconditionality*) and $\perp\!\!\!\perp$ (*perpendicularity*) which are defined on the set of conditionals, as follows (see also [Kern-Isberner, 2001b]):

$$\begin{array}{lll} (D|C) \sqsubseteq (B|A) & \text{iff} & CD \models AB \text{ and } C\overline{D} \models A\overline{B} \\ (D|C) \perp\!\!\!\perp (B|A) & \text{iff} & \text{either } C \models AB, \\ & & \text{or } C \models A\overline{B}, \text{ or } C \models \overline{A} \end{array}$$

If $(D|C) \sqsubseteq (B|A)$, then $(D|C)$ is a *subconditional* of $(B|A)$; if $(D|C) \perp\!\!\!\perp (B|A)$, then $(D|C)$ is called *perpendicular* to $(B|A)$. Both relations are non-symmetrical. Although they symbolize quite opposite ways of conditional interaction, they both aim at excluding conflicts between the involved conditionals. Subconditionals of a conditional are in complete accordance with that conditional, they only possibly apply to fewer worlds. On the other hand, perpendicularity expresses a kind of irrelevance of one conditional for another one. In our structural framework, it can be characterized, as

follows:

$$(D|C)\perp\!\!\!\perp(B|A) \quad \text{iff} \quad \sigma_{(B|A)}(\omega) = \sigma_{(B|A)}(\omega') \text{ for all } \omega, \omega' \models C$$

The next proposition illustrates a connection between perpendicularity and an irrelevance property of the structural inference relation $\mid\!\sim^s_{\mathcal{R}}$.

Proposition 11 *Let C, D, E be propositional formulas such that $C \mid\!\sim^s_{\mathcal{R}} D$, and for all $(B|A) \in \mathcal{R}$, it holds that $(E|CD)\perp\!\!\!\perp(B|A)$. Then $CE \mid\!\sim^s_{\mathcal{R}} D$.*

Proof. Let $\succeq_\pi \in \Pi^s(\mathcal{R})$, arbitrary. Then $CD \succ_\pi C\overline{D}$, since $C \mid\!\sim^s_{\mathcal{R}} D$. In particular, for each $\omega \models CE\overline{D}$, there is $\omega_0 \models CD$ such that $\omega_0 \succ_\pi \omega$. By presupposition, $(E|CD)$ is perpendicular to any conditional in $\mathcal{R}$, so for all $\omega' \models CD$, $\sigma_{\mathcal{R}}(\omega') = \sigma_{\mathcal{R}}(\omega_0)$. From the s-compatibility of $\succeq_\pi$, we conclude $\omega' \approx_\pi \omega_0$, and hence $\omega' \succ_\pi \omega$, in particular, for all $\omega' \models CED$. Therefore, we have $CE \mid\!\sim^s_{\mathcal{R}} D$. □

Moreover, based on the two relations $\sqsubseteq$ and $\perp\!\!\!\perp$, a sufficient condition for a defeasible consequence from $\mathcal{R}$ can be given:

Proposition 12 *Let $C, D \in \mathcal{L}$ be propositional formulas such that for all $(B|A) \in \mathcal{R}$, it holds that either $(D|C) \sqsubseteq (B|A)$, or $(D|C)\perp\!\!\!\perp(B|A)$, and there is at least one conditional $(B|A) \in \mathcal{R}$ such that $(D|C) \sqsubseteq (B|A)$. Then $C \mid\!\sim^s_{\mathcal{R}} D$.*

Proof. Let $\mathcal{R} = \{(B_1|A_1), \ldots, (B_n|A_n)\}$. Without loss of generality, we assume $(D|C) \sqsubseteq (B_1|A_1), \ldots, (B_m|A_m)$, and $(D|C)\perp\!\!\!\perp(B_{m+1}|A_{m+1}), \ldots, (B_n|A_n)$, where $1 \leqslant m \leqslant n$. Then for any $\omega \models CD$, $\omega' \models C\overline{D}$, we have $sign_i(\sigma_{\mathcal{R}}(\omega)) = +, sign_i(\sigma_{\mathcal{R}}(\omega')) = -$, for $1 \leqslant i \leqslant m$, and $sign_i(\sigma_{\mathcal{R}}(\omega)) = sign_i(\sigma_{\mathcal{R}}(\omega'))$ for $m < i \leqslant n$. Hence $\sigma_{\mathcal{R}}(\omega) > \sigma_{\mathcal{R}}(\omega')$, and Proposition 4 yields $C \mid\!\sim^s_{\mathcal{R}} D$. □

In summary, the structural inference relation $\mid\!\sim^s_{\mathcal{R}}$ satisfies all of the postulates listed in Section 1, except for Rational Monotonicity. It must be emphasized that, although s-compatible comparative possibility distributions are only superficially linked to conditional structures, the concept of conditional structures is strong enough to establish a surprising well-behavedness of the corresponding inference relation. There is, however, an even more powerful application of conditional structures for possibility distributions, which will be developed in the following section.

5 Default reasoning based on conditional structures

In this section, we will leave the purely qualitative framework and deal with possibility distributions instead of the induced orderings. Although we are still mainly interested in qualitative relationships, calculating with numbers will make it easier to deal with complex interactions of conditionals. It is straightforward to define s-compatibility with a set of conditionals $\mathcal{R}$ (see Definition 3) for possibility distributions, too, but we will now go one step further and focus on possibility distributions which can be considered as numerical realizations of conditional structures – each of the generators $\mathbf{a}_i^+, \mathbf{a}_i^-$ of $\mathcal{F}_{\mathcal{R}}$ is mapped to a positive real number α_i^+, α_i^-, respectively, in an appropriate way to build up a possitibility distribution representing $\mathcal{R}$. Such possibility distributions will be called *c-representations*.

Definition 13 A positive possibility distribution π is called a *c-representation* of $\mathcal{R} = \{(B_1|A_1), \ldots, (B_n|A_n)\}$ iff $\pi \models \mathcal{R}$ and there are positive real numbers $\alpha_0, \alpha_1^+, \alpha_1^-, \ldots, \alpha_n^+, \alpha_n^- \in \mathbb{R}^+$ such that for all $\omega \in \Omega$,

$$\pi(\omega) = \alpha_0 \prod_{\substack{1 \leqslant i \leqslant n \\ \omega \models A_iB_i}} \alpha_i^+ \prod_{\substack{1 \leqslant i \leqslant n \\ \omega \models A_i\overline{B}_i}} \alpha_i^- \qquad (2)$$

The α_0 in (2) is only a normalization factor to ensure that $\pi(\top) = 1$. The representation property of a distribution of type (2) can be expressed directly by the factors $\alpha_1^+, \alpha_1^-, \ldots, \alpha_n^+, \alpha_n^-$:

Proposition 14 *A possibility distribution $\pi(\omega) = \alpha_0 \prod_{\substack{1 \leqslant i \leqslant n \\ \omega \models A_iB_i}} \alpha_i^+ \prod_{\substack{1 \leqslant i \leqslant n \\ \omega \models A_i\overline{B}_i}} \alpha_i^-$ with $\alpha_0, \alpha_1^+, \alpha_1^-, \ldots, \alpha_n^+, \alpha_n^- \in \mathbb{R}^+$ represents $\mathcal{R} = \{(B_1|A_1), \ldots, (B_n|A_n)\}$ iff for all $i, 1 \leqslant i \leqslant n$,*

$$\alpha_i^+ \cdot \max_{\omega \models A_jB_j} \prod_{\substack{j \neq i \\ \omega \models A_jB_j}} \alpha_j^+ \prod_{\substack{j \neq i \\ \omega \models A_j\overline{B}_j}} \alpha_j^- \qquad (3)$$

$$> \quad \alpha_i^- \cdot \max_{\omega \models A_i\overline{B}_i} \prod_{\substack{j \neq i \\ \omega \models A_jB_j}} \alpha_j^+ \prod_{\substack{j \neq i \\ \omega \models A_j\overline{B}_j}} \alpha_j^-$$

Actually, (3) is equivalent to $\pi(A_iB_i) > \pi(A_i\overline{B}_i)$, $1 \leqslant i \leqslant n$. We will call a mapping $I : \{\mathbf{a}_1^+, \mathbf{a}_1^-, \ldots, \mathbf{a}_n^+, \mathbf{a}_n^-\} \to \mathbb{R}^+, \mathbf{a}_i^+ \mapsto \alpha_i^+, \mathbf{a}_i^- \mapsto \alpha_i^-, 1 \leqslant i \leqslant n$, such that (3) holds, a *c-interpretation of $\mathcal{R}$*. From c-interpretations, c-representations arise in a straightforward way via the formal tool of conditional structures.

The structure imposed on possible worlds by a c-representation is as finely grained as can be justified by differences in conditional structures. More precisely, c-representations are *conditionally indifferent with respect to* $\mathcal{R}$ in the following sense: For any two multi-sets of worlds, $\{\omega_1 : r_1, \ldots, \omega_m : r_m\}$ and $\{\nu_1 : s_1, \ldots, \nu_p : s_p\}$[2] with the same number of elements $\sum_{j=1}^{m} r_j = \sum_{k=1}^{p} s_k$, it holds that $\pi(\omega_1)^{r_1} \ldots \pi(\omega_m)^{r_m} = \pi(\nu_1)^{s_1} \ldots \pi(\nu_p)^{s_p}$ whenever $\sigma_{\mathcal{R}}(\omega_1)^{r_1} \ldots \sigma_{\mathcal{R}}(\omega_m)^{r_m} = \sigma_{\mathcal{R}}(\nu_1)^{s_1} \ldots \sigma_{\mathcal{R}}(\nu_p)^{s_p}$. This means that, in following conditional structures, c-representations handle interactions of conditionals in a most adequate way. For a more thorough discussion of conditional indifference, cf. [Kern-Isberner, 2001a; 2001b; 2001d; 2001c].

Let $\mid\!\sim^{c}_{\mathcal{R}}$ be the (nonmonotonic) inference relation based on considering all c-representations:

$$A \mid\!\sim^{c}_{\mathcal{R}} B \quad \text{iff} \quad \pi_c(AB) > \pi_c(A\overline{B}) \text{ for all c-representations } \pi_c \text{ of } \mathcal{R},$$

where $\mathcal{R} \subseteq (\mathcal{L} \mid \mathcal{L}), A, B \in \mathcal{L}$.

It is easily proved that $\mid\!\sim^{c}_{\mathcal{R}}$ also satisfies the basic properties of system P, *irrelevance*, *specificity*, *property inheritance*, *ambiguity preservation* and *duplication stability*, just in the same way as $\mid\!\sim^{s}_{\mathcal{R}}$ does. In fact, Propositions 6 and 9 can be rephrased for $\mid\!\sim^{c}_{\mathcal{R}}$, and all examples of Section 4 work for $\mid\!\sim^{s}_{\mathcal{R}}$ as well. Furthermore, in the next example, also illustrating *property inheritance*, we will show how inference based on c-representations yields more plausible conclusions than that based on s-representations. The reason for this is that equations (3) often imply qualitative relations between the $\alpha_1^+, \alpha_1^-, \ldots, \alpha_n^+, \alpha_n^-$ which can be used for inferences.

Example 15 Let $\mathcal{R}$ consist of the following conditionals:

$r_1 : (w\|p)$	*Penguins have wings.*
$r_2 : (f\|w)$	*Winged animals fly.*
$r_3 : (\overline{f}\|p)$	*Penguins do not fly.*
$r_4 : (b\|p)$	*Penguins are birds.*
$r_5 : (e\|b)$	*Birds lay eggs.*

In Table 1, we list the conditional structures of all possible worlds with respect to $\mathcal{R}$. Possibility values of c-representations π_c can be calculated directly from that table. For instance, $\pi_c(pw\overline{f}be) = \alpha_0 \alpha_1^+ \alpha_2^- \alpha_3^+ \alpha_4^+ \alpha_5^+$, where the α's are suitably chosen so as to fulfill (3) for that special set of conditionals. For α_4^+, α_4^- and

[2]The numbers r_j, s_k denote the number of copies of the corresponding world to be contained in the multi-set.

ω	$\sigma_{\mathcal{R}}(\omega)$	ω	$\sigma_{\mathcal{R}}(\omega)$
$pwfbe$	$a_1^+ a_2^+ a_3^- a_4^+ a_5^+$	$\overline{p}wfbe$	$a_2^+ a_5^+$
$pwfb\overline{e}$	$a_1^+ a_2^+ a_3^- a_4^+ a_5^-$	$\overline{p}wfb\overline{e}$	$a_2^+ a_5^-$
$pwf\overline{b}e$	$a_1^+ a_2^+ a_3^- a_4^-$	$\overline{p}wf\overline{b}e$	a_2^+
$pwf\overline{b}\overline{e}$	$a_1^+ a_2^+ a_3^- a_4^-$	$\overline{p}wf\overline{b}\overline{e}$	a_2^+
$pw\overline{f}be$	$a_1^+ a_2^- a_3^+ a_4^+ a_5^+$	$\overline{p}w\overline{f}be$	$a_2^- a_5^+$
$pw\overline{f}b\overline{e}$	$a_1^+ a_2^- a_3^+ a_4^+ a_5^-$	$\overline{p}w\overline{f}b\overline{e}$	$a_2^- a_5^-$
$pw\overline{f}\,\overline{b}e$	$a_1^+ a_2^- a_3^+ a_4^-$	$\overline{p}w\overline{f}\,\overline{b}e$	a_2^-
$pw\overline{f}\,\overline{b}\overline{e}$	$a_1^+ a_2^- a_3^+ a_4^-$	$\overline{p}w\overline{f}\,\overline{b}\overline{e}$	a_2^-
$p\overline{w}fbe$	$a_1^- a_3^- a_4^+ a_5^+$	$\overline{p}\,\overline{w}fbe$	a_5^+
$p\overline{w}fb\overline{e}$	$a_1^- a_3^- a_4^+ a_5^-$	$\overline{p}\,\overline{w}fb\overline{e}$	a_5^-
$p\overline{w}f\overline{b}e$	$a_1^- a_3^- a_4^-$	$\overline{p}\,\overline{w}f\overline{b}e$	1
$p\overline{w}f\overline{b}\overline{e}$	$a_1^- a_3^- a_4^-$	$\overline{p}\,\overline{w}f\overline{b}\overline{e}$	1
$p\overline{w}\overline{f}be$	$a_1^- a_3^+ a_4^+ a_5^+$	$\overline{p}\,\overline{w}\overline{f}be$	a_5^+
$p\overline{w}\overline{f}b\overline{e}$	$a_1^- a_3^+ a_4^+ a_5^-$	$\overline{p}\,\overline{w}\overline{f}b\overline{e}$	a_5^-
$p\overline{w}\overline{f}\,\overline{b}e$	$a_1^- a_3^+ a_4^-$	$\overline{p}\,\overline{w}\overline{f}\,\overline{b}e$	1
$p\overline{w}\overline{f}\,\overline{b}\overline{e}$	$a_1^- a_3^+ a_4^-$	$\overline{p}\,\overline{w}\overline{f}\,\overline{b}\overline{e}$	1

Table 1: Conditional structures for Example 15

α_5^+, α_5^-, e.g., we obtain from (3)

$$\begin{aligned}
&\alpha_4^+ \cdot \max\{\alpha_1^+\alpha_2^+\alpha_3^-\alpha_5^+, \alpha_1^+\alpha_2^+\alpha_3^-\alpha_5^-, \alpha_1^+\alpha_2^-\alpha_3^+\alpha_5^+, \\
&\qquad \alpha_1^+\alpha_2^-\alpha_3^+\alpha_5^-, \alpha_1^-\alpha_3^-\alpha_5^+, \alpha_1^-\alpha_3^-\alpha_5^-, \\
&\qquad \alpha_1^-\alpha_3^+\alpha_5^+, \alpha_1^-\alpha_3^+\alpha_5^-\} \\
&> \alpha_4^- \cdot \max\{\alpha_1^+\alpha_2^+\alpha_3^-, \alpha_1^+\alpha_2^-\alpha_3^+, \alpha_1^-\alpha_3^-, \alpha_1^-\alpha_3^+\} \\
&\alpha_5^+ \cdot \max\{\alpha_1^+\alpha_2^+\alpha_3^-\alpha_4^+, \alpha_1^+\alpha_2^-\alpha_3^+\alpha_4^+, \alpha_1^-\alpha_3^-\alpha_4^+, \\
&\qquad \alpha_1^-\alpha_3^+\alpha_4^+, \alpha_2^+, \alpha_2^-, 1\} \\
&> \alpha_5^- \cdot \max\{\alpha_1^+\alpha_2^+\alpha_3^-\alpha_4^+, \alpha_1^+\alpha_2^-\alpha_3^+\alpha_4^+, \alpha_1^-\alpha_3^-\alpha_4^+, \\
&\qquad \alpha_1^-\alpha_3^+\alpha_4^+, \alpha_2^+, \alpha_2^-, 1\}
\end{aligned}$$

So, $\alpha_5^+ > \alpha_5^-$, and therefore,

$$\begin{aligned}
&\alpha_4^+\alpha_5^+ \cdot \max\{\alpha_1^+\alpha_2^+\alpha_3^-, \alpha_1^+\alpha_2^-\alpha_3^+, \alpha_1^-\alpha_3^-, \alpha_1^-\alpha_3^+\} \\
&> \alpha_4^- \cdot \max\{\alpha_1^+\alpha_2^+\alpha_3^-, \alpha_1^+\alpha_2^-\alpha_3^+, \alpha_1^-\alpha_3^-, \alpha_1^-\alpha_3^+\},
\end{aligned}$$

which implies $\alpha_4^+\alpha_5^+ > \alpha_4^-$.

Although penguins are exceptional birds in that they are not able to fly, they should still inherit the property of laying eggs from their superclass *birds*. This is not guaranteed in general possibilistic inference – the conditional $(e|b)$ is "drowned" by the conditionals $(w|p), (\overline{f}|p)$ which obtain a higher degree of possibility (see [Benferhat *et al.*, 1997, p. 273]). Note that also no application of Proposition 4 is possible to yield an easy $\mid\!\sim^{s}_{\mathcal{R}}$-inference, since $\sigma_{\mathcal{R}}(p\overline{e})$ contain some conditional structures which are not dominated by conditional structures in $\sigma_{\mathcal{R}}(pe)$. By following conditional

structures as in c-representations, however, the impact of $(e|b)$ can be preserved to establish the validity of $(e|p)$: For any c-representation π_c of $\mathcal{R}$, we compute

$$\begin{aligned}\pi_c(pe) &= \alpha_0 \max\{\alpha_1^+\alpha_2^+\alpha_3^-\alpha_4^+\alpha_5^+, \alpha_1^+\alpha_2^+\alpha_3^-\alpha_4^-,\\ &\quad \alpha_1^+\alpha_2^-\alpha_3^+\alpha_4^+\alpha_5^+, \alpha_1^+\alpha_2^-\alpha_3^+\alpha_4^-,\\ &\quad \alpha_1^-\alpha_3^-\alpha_4^+\alpha_5^+, \alpha_1^-\alpha_3^-\alpha_4^-,\\ &\quad \alpha_1^-\alpha_3^+\alpha_4^+\alpha_5^+, \alpha_1^-\alpha_3^+\alpha_4^-\}\\ \pi_c(p\overline{e}) &= \alpha_0 \max\{\alpha_1^+\alpha_2^+\alpha_3^-\alpha_4^+\alpha_5^-, \alpha_1^+\alpha_2^+\alpha_3^-\alpha_4^-,\\ &\quad \alpha_1^+\alpha_2^-\alpha_3^+\alpha_4^+\alpha_5^-, \alpha_1^+\alpha_2^-\alpha_3^+\alpha_4^-,\\ &\quad \alpha_1^-\alpha_3^-\alpha_4^+\alpha_5^-, \alpha_1^-\alpha_3^-\alpha_4^-,\\ &\quad \alpha_1^-\alpha_3^+\alpha_4^+\alpha_5^-, \alpha_1^-\alpha_3^+\alpha_4^-\}\end{aligned}$$

By comparing the sets which contribute to the maxima element by element, we see that $\pi_c(pe) > \pi_c(p\overline{e})$, due to $\alpha_5^+ > \alpha_5^-$ and $\alpha_4^+\alpha_5^+ > \alpha_4^-$ (e.g., $\alpha_1^+\alpha_2^+\alpha_3^-\alpha_4^+\alpha_5^+ > \alpha_1^+\alpha_2^+\alpha_3^-\alpha_4^-$ and $\alpha_1^+\alpha_2^+\alpha_3^-\alpha_4^+\alpha_5^+ > \alpha_1^+\alpha_2^+\alpha_3^-\alpha_4^+\alpha_5^-$). This means, that $\pi_c \models (e|p)$, for all c-representations π_c of $\mathcal{R}$. Hence, $p \mathrel{|\!\sim^c_{\mathcal{R}}} e$. ■

In this way, c-representations combine structural with qualitative information to handle conditional interactions.

Our general approach based on conditional structures and c-representations may be simplified when combined with the *principle of least commitment* and with the *principle of minimum specificity*, that is, *choose the α's in a least committed and least specific way.* The consequences of these principles are twofold:

1. To allow worlds verifying conditionals to be as normal and least specific as possible, set $\alpha_i^+ = 1$ for $1 \leqslant i \leqslant n$. Since $\mathcal{R}$ is consistent iff there are worlds not falsifying any of the conditionals in $\mathcal{R}$, this implies $\alpha_0 = 1$. Furthermore, to comply with intuition (falsifying conditionals should decrease the possibility of worlds), we choose $\alpha_i^- < 1$ for $1 \leqslant i \leqslant n$.
2. In order not to introduce any relations which are not explicitly sanctioned by the representation postulate $\pi \models \mathcal{R}$, choose $\alpha_1^-, \ldots, \alpha_n^-$ according to (3), but in a "least committed way". Since only qualitative relationships count, this comes down to the following: The set $\{\alpha_1^-, \ldots, \alpha_n^-\}$ is partitioned into sets $\mathcal{A}_1, \ldots, \mathcal{A}_m$ such that the number m of partitioning sets is minimal, each α_i^- is in a partitioning set $\mathcal{A}_j$ with an index as minimal as possible, and the partitioning is compatible with the ordering between $\alpha_1^-, \ldots, \alpha_n^-$, in that if $\alpha_{i_1}^- \in \mathcal{A}_j$ and $\alpha_{i_2}^- \in \mathcal{A}_k$ with $j < k$, then $\alpha_{i_1}^- > \alpha_{i_2}^-$.

An *lcc-representation* of $\mathcal{R}$ is a c-representation $\pi_{lcc}(\omega) = \prod_{\substack{1\leqslant i\leqslant n\\ \omega\models A_i\overline{B}_i}} \alpha_i^-$ of $\mathcal{R}$ whose factors α_i^- satisfy both requirements 1. and 2. above. In particular, $\alpha_i^- < 1$, and

$$\max_{\omega\models A_iB_i} \prod_{\substack{j\neq i\\ \omega\models A_j\overline{B}_j}} \alpha_j^- > \alpha_i^- \cdot \max_{\omega\models A_i\overline{B}_i} \prod_{\substack{j\neq i\\ \omega\models A_j\overline{B}_j}} \alpha_j^-,$$

$1 \leqslant i \leqslant n$, for lcc-representations π_{lcc}. Let $\mathrel{|\!\sim^{lcc}_{\mathcal{R}}}$ be the inference relation based on considering all lcc-representations:

$$A \mathrel{|\!\sim^{lcc}_{\mathcal{R}}} B \quad \text{iff} \quad \pi_{lcc}(AB) > \pi_{lcc}(A\overline{B}) \text{ for all lcc-representations } \pi_{lcc} \text{ of } \mathcal{R}.$$

The approach we thereby obtain is very similar to the *lcd-approach* of Benferhat, Saffiotti and Smets [Benferhat *et al.*, 2000]. Instead of *proto-infinitesimals* and *infinitesimals* which are used in that paper, we use formal symbols $\mathrm{a}_1^-, \ldots, \mathrm{a}_n^-$ and positive real numbers $\alpha_1^-, \ldots, \alpha_n^-$. So our c-interpretations are nearly the same as their interpretations of proto-infinitesimals. Their *auto-deduction principle* is the same as our (quite implicit) representation postulate $\pi \models \mathcal{R}$. They use belief functions, but actually base their considerations upon the corresponding plausibility functions; we use possibility distributions (which correspond to plausibility functions) directly. The basic idea to construct infinitesimal lcd-belief functions is to interpret proto-infinitesimals according to a *least committed ϵ-stratification*, where each ϵ-stratification corresponds to a partition of $\mathcal{R}$, and least committedness comes down to minimizing the number of classes, and to assign as many proto-infinitesimals (i.e. α_i^- in our framework) as possible to classes with low index. This amounts exactly to choose $\alpha_1^-, \ldots, \alpha_n^-$ according to (3) in a least committed way, as described above in 2. So our $\mathrel{|\!\sim^{lcc}_{\mathcal{R}}}$ and $\mathrel{|\!\sim_{lcd}}$ of [Benferhat *et al.*, 2000] coincide.

When comparing this present paper to [Benferhat *et al.*, 2000], it must be emphasized that both papers agree on the ideas of taking each default rule as an independent piece of information and of aggregating the effects of default rules in a mainly structural way. The crucial difference between our work and that of Benferhat, Saffiotti and Smets [Benferhat *et al.*, 2000] is how these ideas are realized, in particular, how the special product form of possibility distributions (see (2)), or belief functions in [Benferhat *et al.*, 2000], respectively, is motivated: Benferhat, Saffiotti and Smets derive the product form (of the corresponding plausibility functions) from Dempster's combination rule, whereas in our framework, (2) arises from an underlying conditional theory, which is much more

general and applies not only in a possibilistic environment, but also in other quantitative (probabilistic) and semi-quantitative (ordinal conditional functions) environments and has also important consequences for handling conditionals adequately in a purely qualitative manner (cf. [Kern-Isberner, 2001b]). Actually, in the framework of ordinal conditional functions, the application of our conditional approach turns out to be a generalization of system-Z^+ (cf. [Kern-Isberner, 2001c]).

6 Conclusion

In this paper, we presented several approaches to default reasoning in a possibilistic framework which all are based on considering *conditional structures* of worlds. Conditional structures encode the effects conditionals have on worlds and make conditional interactions transparent and computable, providing a sound algebraic framework for default inferences. Even the weakest of our inference relations, $\mid\!\sim^{s}_{\mathcal{R}}$, which merely demands a compatibility with conditional structures, proved to be quite well-behaved, according to widely accepted standards. The other two inference relations investigated here, $\mid\!\sim^{c}_{\mathcal{R}}$ and $\mid\!\sim^{lcc}_{\mathcal{R}}$, are based on faithful possibilistic realizations of conditional structures. Both these approaches allow to handle implicit priorities among the default rules under consideration and thereby combine structural with qualitative information. $\mid\!\sim^{c}_{\mathcal{R}}$ reflects conditional structures in full generality, whereas $\mid\!\sim^{lcc}_{\mathcal{R}}$ (which turns out to be equivalent to $\mid\!\sim_{lcd}$ of [Benferhat *et al.*, 2000]) is also guided by postulates of least specificity and least committedness.

The concept of conditional structures was initiated when investigating maximum entropy inference (cf. [Kern-Isberner, 1998]). The same crucial principle, namely that of observing conditional structures, underlies the possibilistic approaches to default reasoning presented in this paper, as well as the probabilistic maximum entropy approach, the generalized system-Z^+ [Goldszmidt *et al.*, 1993; Kern-Isberner, 2001c] and the lcd-approach of Benferhat, Saffiotti and Smets [Benferhat *et al.*, 2000]. Independent of numerical realizations, however, this paper reveals clearly the relevance of structural arguments in default reasoning.

References

[Adams, 1975] E.W. Adams. *The Logic of Conditionals*. D. Reidel, Dordrecht, 1975.

[Benferhat *et al.*, 1992] S. Benferhat, D. Dubois, and H. Prade. Representing default rules in possibilistic logic. In *Proceedings 3th International Conference on Principles of Knowledge Representation and Reasoning KR'92*, pages 673–684, 1992.

[Benferhat *et al.*, 1997] S. Benferhat, D. Dubois, and H. Prade. Nonmonotonic reasoning, conditional objects and possibility theory. *Artificial Intelligence*, 92:259–276, 1997.

[Benferhat *et al.*, 1999] S. Benferhat, D. Dubois, and H. Prade. Possibilistic and standard probabilistic semantics of conditional knowledge bases. *Journal of Logic and Computation*, 9(6):873–895, 1999.

[Benferhat *et al.*, 2000] S. Benferhat, A. Saffiotti, and P. Smets. Belief functions and default reasoning. *Artificial Intelligence*, 122:1–69, 2000.

[Calabrese, 1991] P.G. Calabrese. Deduction and inference using conditional logic and probability. In I.R. Goodman, M.M. Gupta, H.T. Nguyen, and G.S. Rogers, editors, *Conditional Logic in Expert Systems*, pages 71–100. Elsevier, North Holland, 1991.

[DeFinetti, 1974] B. DeFinetti. *Theory of Probability*, volume 1,2. John Wiley and Sons, New York, 1974.

[Dubois and Prade, 1996] D. Dubois and H. Prade. Non-standard theories of uncertainty in plausible reasoning. In G. Brewka, editor, *Principles of Knowledge Representation*. CSLI Publications, 1996.

[Dubois *et al.*, 1994] D. Dubois, J. Lang, and H. Prade. Possibilistic logic. In D.M. Gabbay, C.H. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 439–513. Oxford University Press, 1994.

[Fine and Rosenberger, 1999] B. Fine and G. Rosenberger. *Algebraic Generalizations of Discrete Groups*. Dekker, New York, Basel, 1999.

[Geffner and Pearl, 1992] H. Geffner and J. Pearl. Conditional entailment: bridging two approaches to default reasoning. *Artificial Intelligence*, 53:209–244, 1992.

[Geffner, 1992] H. Geffner. *Default Reasoning: Causal and Conditional Theories*. MIT Press, Cambridge, Mass., 1992.

[Goldszmidt and Pearl, 1996] M. Goldszmidt and J. Pearl. Qualitative probabilities for default reasoning, belief revision, and causal modeling. *Artificial Intelligence*, 84:57–112, 1996.

[Goldszmidt *et al.*, 1993] M. Goldszmidt, P. Morris, and J. Pearl. A maximum entropy approach to nonmonotonic reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):220–232, 1993.

[Kern-Isberner, 1998] G. Kern-Isberner. Characterizing the principle of minimum cross-entropy within a conditional-logical framework. *Artificial Intelligence*, 98:169–208, 1998.

[Kern-Isberner, 2000] G. Kern-Isberner. Solving the inverse representation problem. In *Proceedings 14th European Conference on Artificial Intelligence, ECAI'2000*, pages 581–585, Berlin, 2000. IOS Press.

[Kern-Isberner, 2001a] G. Kern-Isberner. Conditional preservation and conditional indifference. *Journal of Applied Non-Classical Logics*, 11(1-2):85–106, 2001.

[Kern-Isberner, 2001b] G. Kern-Isberner. *Conditionals in nonmonotonic reasoning and belief revision.* Springer, Lecture Notes in Artificial Intelligence LNAI 2087, 2001.

[Kern-Isberner, 2001c] G. Kern-Isberner. Handling conditionals adequately in uncertain reasoning. In *Proceedings European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'01*, pages 604–615. Springer LNAI 2143, 2001.

[Kern-Isberner, 2001d] G. Kern-Isberner. Representing and learning conditional information in possibility theory. In *Proceedings 7th Fuzzy Days, Dortmund, Germany*, pages 194–217. Springer LNCS 2206, 2001.

[Kraus *et al.*, 1990] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.

[Lehmann and Magidor, 1992] D. Lehmann and M. Magidor. What does a conditional knowledge base entail? *Artificial Intelligence*, 55:1–60, 1992.

[Lehmann, 1995] D. Lehmann. Another perspective on default reasoning. *Ann. Math. Artificial Intelligence*, 15:61–82, 1995.

[Lyndon and Schupp, 1977] R.C. Lyndon and P.E. Schupp. *Combinatorial group theory.* Springer, Berlin Heidelberg New York, 1977.

[Reiter, 1980] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

Qualitative Choice Logic

Gerhard Brewka
Universität Leipzig
Institut für Informatik
Augustusplatz 10-11
D-04109 Leipzig, Germany
brewka@informatik.uni-leipzig.de

Salem Benferhat
IRIT - CNRS
Université Paul Sabatier
118 Route de Narbonne
31062 Toulouse Cedex 4, France
benferhat@irit.fr

Daniel Le Berre
CRIL - CNRS
Université d'Artois
Rue Jean Souvraz SP 18
62307 Lens Cedex, France
leberre@cril.univ-artois.fr

Abstract

Qualitative choice logic (QCL) is a propositional logic for representing alternative, ranked options for problem solutions. The logic adds to classical propositional logic a new connective called ordered disjunction: $A \times B$ intuitively means: if possible A, but if A is not possible then at least B. The semantics of qualitative choice logic is based on a preference relation among models. Consequences of QCL theories can be computed through a compilation to stratified knowledge bases which in turn, according to results in [2], can be compiled to classical propositional theories.

1 Introduction

For many AI applications, e.g. in design or configuration, it is necessary to represent intended properties of a particular problem solution. For instance, if we want to make a hotel booking for a trip to a conference, we intend properties such as being close to the conference site, being close to potential sight-seeing objects, and we want the hotel reasonably priced. Most often not all of the intended properties can be satisfied, that is, we have to make some sort of compromises. To do so it is very convenient to be able to express alternative, second (or third etc.) best options which one would like satisfied if the best option is unavailable. For the hotel booking example, for instance, we may want to express that we prefer to stay within walking distance of the conference site, if that is not possible transportation provided by the hotel should be available, if also this is not possible we want at least public transportation (taxis are not being reimbursed according to our university's travel refund policy).

To represent options of this kind we introduce in this paper a new nonmonotonic propositional logic for representing qualitative choices - hence the name qualitative choice logic (QCL). The logic is different from existing nonmonotonic logics in the way nonmonotonicity is introduced: we do not use non-standard inference rules, as in Reiter's default logic [22], modal operators expressing consistency or belief, as in autoepistemic logic [20], or abnormality predicates whose extensions are minimized, as in circumscription [18, 19]. The nonstandard part of our logic is a new logical connective $\times$ which is fully embedded in the logical language. Intuitively, if A and B are formulas then $A \times B$ says: if possible A, but if A is impossible then (at least) B.

The intended use of this logic can be illustrated using the hotel booking example. Assume we want to represent the options concerning the location. Using mnemonic symbols we express the options as follows:

$$walking \times hotel\text{-}transport \times public\text{-}transport$$

Assume there are 4 hotels available out of which we have to pick one:

$$hotel_1 \vee hotel_2 \vee hotel_3 \vee hotel_4$$

We have the following information about the hotels

$$\begin{aligned} hotel_1 &\rightarrow walking \\ hotel_2 &\rightarrow \neg walking \wedge hotel\text{-}transport \\ hotel_3 &\rightarrow \neg walking \wedge \neg hotel\text{-}transport \\ &\quad \wedge public\text{-}transport \\ hotel_4 &\rightarrow \neg walking \wedge \neg hotel\text{-}transport \\ &\quad \wedge \neg public\text{-}transport \end{aligned}$$

Given these propositional formulas QCL will give us the conclusion $hotel_1$ since this is the only hotel satisfying our most intended option. Now assume that, after calling the hotel we find out that it is fully booked,

that is, we have the additional information $\neg hotel_1$. This means that our most favoured property, being within walking distance of the conference site, cannot be satisfied. We now obtain the conclusion $hotel_2$ which is not exactly what we wanted but better than nothing.

Our new connective $\times$ can be viewed as a kind of disjunction where the order of the disjuncts matters: $A \times B$ is obviously very different form $B \times A$. We therefore call it ordered disjunction. Its semantics will be defined in terms of preferred models.

The rest of the paper is organized as follows: we first introduce syntax and semantics for *QCL* and give a few motivating examples. We then consider aspects of computation. It turns out that formulas have a special normal form. Theories in this normal form can be translated to stratified knowledge bases. These in turn can, according to the results in [2], be transformed into a classical propositional theory. We thus can compute consequences of *QCL* through a compilation process. Sect. 4 describes potential applications of *QCL*. In Sect. 5 we present several alternative definitions of the consequence relation. Sect. 6 concludes the paper. Proofs of propositions are contained in the Appendix.

2 Syntax and semantics of *QCL*

2.1 Syntax

We start with standard propositional logic and add a new non-standard kind of disjunction: given formulas $A_1, \ldots, A_n$[1] we use

$$A_1 \times \ldots \times A_n$$

to express: some A_j must be true, preferably A_1, but if this is not possible then A_2, if this is not possible A_3, etc. The idea is that from $A \times B$ you get A, from $A \times B, \neg A$ you get B ($A \times B, \neg A, \neg B$ is inconsistent). Clearly, the order is what matters. We, therefore, call $\times$ ordered disjunction.

Ordered disjunction is fully embedded in the logical language to make sure that context dependent options can be expressed, that is we may have formulas like $A \rightarrow (B \times C)$ and $\neg A \rightarrow (C \times B)$. The precise definition of the syntax is as follows:

Definition 1 *Let $\mathcal{V}$ be a set of atoms. The set of well-formed formulas of QCL is inductively defined as follows*

1. *every element of $\mathcal{V}$ is a well-formed formula,*
2. *if F_1 and F_2 are well-formed formulas, then $(\neg F_1)$, $(F_1 \vee F_2)$, $(F_1 \wedge F_2)$ and $(F_1 \times F_2)$ are well-formed formulas.*

As usual we use $A \rightarrow B$ as an abbreviation for $\neg A \vee B$ and $A \leftrightarrow B$ as an abbreviation for $(A \rightarrow B) \wedge (B \rightarrow A)$. We omit unnecessary brackets. $A_1 \times A_2 \times \ldots \times A_n$ abbreviates $(A_1 \times (A_2 \times (\ldots (A_{n-1} \times A_n) \ldots)))$[2].

2.2 Semantics

The semantics of *QCL* is based on the degree of satisfaction of a formula in a particular (classical) model. Intuitively, the degree can be viewed as a measure of disappointment: the higher the degree the more disappointing the model (or the farther away from complete satisfaction).

As in standard propositional logic, an interpretation I is an assignment of the classical truth values T, F to the atoms. We identify I with the subset of true atoms.

Interpretations can satisfy formulas to a certain degree. For instance, if A and B are atoms and I contains A then I satisfies the formula $A \times B$ to degree 1. If I does not contain A but B, then it satisfies $A \times B$ to degree 2.

To determine the degree of disappointment for a model we need to establish, for each formula F, how many different possible levels of satisfaction the formula provides, in other words, how many potential backups (including the most favourable option) F admits. This is captured in the following notion of optionality.

Definition 2 *The optionality of a formula is defined as follows:*

$opt(A) = 1$ *if A is an atom*
$opt(\neg F) = 1$
$opt(F_1 \vee F_2) = max(opt(F_1), opt(F_2))$
$opt(F_1 \wedge F_2) = max(opt(F_1), opt(F_2))$
$opt(F_1 \times F_2) = opt(F_1) + opt(F_2)$

Intuitively, if the optionality of F is n, then there is a best way, a second best way, etc. and an n-th best way of satisfying F. For classical formulas there is only one way to satisfy them, hence they all have optionality 1. If $F = A \times B$, where A and B are classical, 2 degrees are needed. The best way of satisfying F is making A true, but if this doesn't work there is still the second best option, namely making B true.

[1] Throughout the paper we use capital letters from the beginning of the alphabet to represent formulas.

[2] We later show that $\times$ is associative, so the brackets do not matter.

The optionality of a negated formula may seem puzzling at first, but there is not more than one way of making $\neg(A \times B)$ true: you must make A and B false, there is no second best solution for this. In a sense negation transforms nested ordered disjunctions into standard disjunctions. Hence $\neg F$ behaves like a classical formula which explains why $opt(\neg F) = 1$.

Note that for formulas of the form $A_1 \times \ldots \times A_n$ where all A_i are classical the optionality is n.

We next define the satisfaction relation. The relation is indexed according to the degree of satisfaction of a formula in a model.

Definition 3

1. $I \models_k A$ *iff* $k = 1$ *and* $A \in I$ *(for propositional atoms A)*
2. $I \models_k P \wedge Q$ *iff* $I \models_m P$ *and* $I \models_n Q$ *and* $k = max(m, n)$
3. $I \models_k P \vee Q$ *iff* $[I \models_k P$ *and for no* $j < k : I \models_j Q]$ *or* $[I \models_k Q$ *and for no* $j < k : I \models_j P]$
4. $I \models_k \neg P$ *iff* $k = 1$ *and for no* m: $I \models_m P$
5. $I \models_k P \times Q$ *iff* $I \models_k P$ *or* $I \models_1 \neg P$, $I \models_m Q$, *and* $k = m + opt(P)$.

Let us illustrate the satisfiability relation using a simple example. Let $F = A \vee (B \times C)$ where A, B and C are atoms. Now all interpretations I containing A as well as those containing B satisfy F with degree 1, that is, we have $I \models_1 F$. Now assume I is an interpretation which contains C but makes A and B false. In this case F is satisfied with degree 2, that is $I \models_2 F$. Interpretations satisfying none of the three atoms do not satisfy F to any degree; they satisfy $\neg F$ with degree 1.

Lemma 1 $I \models_k F$ *and* $I \models_j F$ *implies* $k = j$.

Lemma 2 $I \models_k F$ *implies* $k \leq opt(F)$.

The converse is obviously false. Indeed, let $F = A \vee (B \times C)$ where A, B, C are atoms. Let $I = \{A, B, C\}$. We have $I \models_1 F$, but $opt(F) = max(opt(A), opt(B \times C)) = 2$.

It will be shown later that all formulas containing $\times$ can be transformed equivalently into so-called basic choice formulas of the form $(A_1 \times \ldots \times A_n)$ where the A_i do not contain $\times$. For formulas of this kind clause 5. in the definition of the satisfaction relation can be simplified to

5′. $I \models_k (A_1 \times \ldots \times A_n)$ iff
$I \models_k A_1$ (in which case $k = 1$) or
$I \models_1 \neg A_1$, $n > 1$, and $I \models_{k-1} (A_2 \times \ldots \times A_n)$.

Note that the *opt* function is not needed for basic choice formulas. To see why in the general case the more complicated definition is required consider the formula

$$F = (walking \times hotel{-}transport) \times public{-}transport$$

and take a model I satisfying *public−transport* but neither *walking* nor *hotel−transport*. According to 5′ I satisfies the formula with degree 2, since I satisfies *public−transport* with 1. But this seems unintuitive since *public−transport* is only the 3rd-best option. According to 5 we obtain $I \models_3 F$.

Proposition 1 *Let F be a formula. There is a k such that $I \models_k F$ iff $I \models F^*$ where F^* is obtained from F by replacing each occurrence of $\times$ with standard disjunction ($\models$ is classical propositional satisfaction).*

Definition 4 *Let T be a set of formulas. An interpretation I is a model of T if it satisfies each formula in T to some degree.*

The satisfaction degrees of formulas help us to determine preferred models. There are different ways of doing this. We will use here a lexicographic ordering of models based on the number of formulas satisfied to a particular degree. In Sect 5 alternative preference relations will be discussed. The lexicographic ordering is defined as follows:

Definition 5 *Let $M^k(T)$ denote the subset of formulas of T satisfied by a model M to degree k. A model M_1 is T-preferred over a model M_2 if there is a k such that $|M_1^k(T)| > |M_2^k(T)|$ and for all $j < k$: $|M_1^j(T)| = |M_2^j(T)|$. M is a preferred model of T iff M is a maximally T-preferred model.*

Intuitively, a preferred model of T is a model of T which satisfies the maximal number of best options of choice logic formulas. Note that *QCL* is a conservative extension of classical propositional logic: for a set of formulas T without appearance of $\times$ the preferred models of T coincide with the classical models of T.

We next define a consequence relation based on preferred models. We will restrict our attention here to classical conclusions, that is, the consequence relation we are interested in will be a relation between sets of

formulas and classical formulas. The justification for this restriction lies in the intended use of *QCL*: we want to be able to derive properties describing a problem solution based on background knowledge, knowledge about the case at hand and choice formulas describing intended properties. Classical formulas are the formulas describing the intended problem solutions (e.g. the chosen hotel in the booking example).

Definition 6 *Let T be a set of formulas, and let A be a classical formula. $T \mid\!\sim A$ iff A is satisfied in all preferred models of T.*

This inference relation is obviously nonmonotonic. As an example consider $T = \{A \times B\}$. We have three models $\{A\}, \{A, B\}, \{B\}$ with satisfaction degree 1, 1, 2, respectively. This means that $\{A\}$ and $\{A, B\}$ are maximally preferred and we have $A \times B \mid\!\sim A$.

If we add $\neg A$ then the single model, and thus the single preferred model, is $\{B\}$. We thus obtain $\{A \times B, \neg A\} \mid\!\sim \neg A$. A is no longer a conclusion.

We now show that our inference relation satisfies properties usually considered intended in nonmonotonic reasoning.

Proposition 2 *The inference relation $\mid\!\sim$ satisfies cautious monotony, that is, $T \mid\!\sim A$ and $T \mid\!\sim B$ implies $T \cup \{A\} \mid\!\sim B$.*

Proposition 3 *The inference relation $\mid\!\sim$ satisfies cumulative transitivity, that is, $T \mid\!\sim A$ and $T \cup \{A\} \mid\!\sim B$ implies $T \mid\!\sim B$.*

In addition, we can show that $\mid\!\sim$ is rational, in the sense that it satisfies all rules of System P [13] as well as rational monotony [17].

3 Computation

In this section we investigate ways to compute consequences of *QCL* theories. The basic idea is to take a set of arbitrary formulas T and to proceed in 3 steps:

1. translate T to $Norm(T)$, a strongly equivalent normal form where all formulas containing $\times$ are basic choice formulas (to be defined below),
2. construct a stratified knowledge base $Skb(Norm(T))$ such that a classical formula F is lexicographically entailed by $Skb(Norm(T))$ iff $Norm(T) \mid\!\sim F$,
3. use the techniques developed in [2] to generate $Extract(Skb(Norm(T)))$, a classical propositional knowledge base whose consequences are exactly the formulas lexicographically entailed by $Skb(Norm(T))$.

From the equivalence results described in the next subsections the following proposition is immediate:

Proposition 4 *Let T be a consistent set of formulas. $T \mid\!\sim A$ iff $Extract(Skb(Norm(T))) \vdash A$.*

3.1 Translation to normal form

We first need a notion of equivalence:

Definition 7 *Let F_1, F_2 be formulas. F_1 is strongly equivalent to F_2, denoted $F_1 \doteq F_2$, iff $opt(F_1) = opt(F_2)$ and for all interpretations I and integers k we have $I \models_k F_1$ iff $I \models_k F_2$.*

The optionality of the formulas must be the same in order to guarantee that subformulas can be replaced. For instance, although $A \times A$ and A have the same satisfaction degree in all interpretations $A \times B$ and $A \times A \times B$ clearly have not. Note that for classical formulas without $\times$ all classical logical transformations can be performed, but for those with $\times$ some standard transformations are not valid: if A is classical, then $\neg\neg A$ is strongly equivalent to A, but $\neg\neg(A \times B)$ is not strongly equivalent to $A \times B$ due to the different optionalities of the formulas.

Proposition 5 *Ordered disjunction is associative, that is for arbitrary formulas F_1, F_2 and F_3 we have $((F_1 \times F_2) \times F_3) \doteq (F_1 \times (F_2 \times F_3))$.*

We want to translate *QCL* bases into a normal form consisting only of classical formulas and basic choice formulas which are defined as follows:

Definition 8 *A formula F is a basic choice formula if it is of the form $(A_1 \times \ldots \times A_n)$ where each A_i is a classical propositional formula and $n > 1$.*

For the translation to normal form we need the following strong equivalences:

Proposition 6 *Let $A_i, B_j, C_k, \ldots$ be formulas without $\times$. Then the following strong equivalences hold:*

1. $(A_1 \times \ldots \times A_n) \vee (B_1 \times \ldots \times B_m) \doteq (C_1 \times \ldots \times C_k)$ *where* $k = max(m, n)$ *and* $C_i =$
   $(A_i \vee B_i)$ *if* $i \leq min(m, n)$
   A_i *if* $m < i \leq n$, *and*
   B_i *if* $n < i \leq m$.
   Example: $(A_1 \times A_2) \vee (B_1 \times B_2 \times B_3) \doteq$
   $(A_1 \vee B_1) \times (A_2 \vee B_2) \times B_3$

2. $(A_1 \times \ldots \times A_n) \wedge (B_1 \times \ldots \times B_m) \doteq (C_1 \times \ldots \times C_k)$
   where $k = max(m,n)$ *and* $C_i =$
   $[(A_1 \vee \ldots \vee A_i) \wedge B_i] \vee [A_i \wedge (B_1 \vee \ldots \vee B_i)]$ *if* $i \leq min(m,n)$
   $[(A_1 \vee \ldots \vee A_n) \wedge B_i]$ *if* $n < i \leq m$
   $[A_i \wedge (B_1 \vee \ldots \vee B_m)]$ *if* $m < i \leq n$.
   Example:
   $(A_1 \times A_2) \wedge (B_1 \times B_2 \times B_3) \doteq$
   $A_1 \wedge B_1 \times$
   $[(A_1 \vee A_2) \wedge B_2] \vee [A_2 \wedge (B_1 \vee B_2)] \times$
   $[(A_1 \vee A_2) \wedge B_3]$

3. $\neg(A_1 \times \ldots \times A_n) \doteq \neg(A_1 \vee \ldots \vee A_n)$

Repeated application of these transformation rules moves $\times$ outside (or eliminates it) until we obtain a classical formula or a basic choice formula:

Proposition 7 *Every formula F can be translated to a strongly equivalent formula F' which is either classical or a basic choice formula.*

We next show how *QCL* bases in normal form can be translated to stratified knowledge bases. For the results of the following section we need an additional condition: we say a set T of formulas is standardized iff different basic choice formulas do not possess syntactically *identical* prefixes[3] (they may, of course, be logically equivalent). This condition is added to avoid dealing with multi-sets in the following definitions. Of course, *any* theory T can be transformed into standardized form. This can simply be achieved by replacing identical formulas with some logically equivalent but syntactically different formula. For instance, a standardized form of $T = \{A \times B, A \times C\}$ can be $\{A \wedge A \times B, A \times C\}$.

A set of *QCL* formulas T is in normal form if it consists of classical or basic choice formulas only, and if it is standardized.

3.2 Compilation to a stratified knowledge base

We recall the definition of a stratified knowledge base.

Definition 9 *A stratified knowledge base is a sequence $(K, S_1, \ldots, S_n)$ of sets of classical propositional formulas.*

Since we need K to be a non-default level we slightly modify the definition of lexicographically preferred subbases from [1, 16].

[3] $A_1 \times \ldots \times A_k$ is a prefix of $A_1 \times \ldots \times A_n$ whenever $k \leq n$.

Definition 10 *Let $KB = (K, S_1, \ldots, S_n)$ be a stratified knowledge base. A maximal consistent subset S of $(K \cup S_1 \cup \ldots \cup S_n)$ is a lexicographically preferred subbase of KB iff*

1. *$K \subseteq S$, and*
2. *if S' is a maximal consistent subset of $(K \cup S_1 \cup \ldots \cup S_n)$ containing K and for some $k \in \{1, \ldots, n\} : |S' \cap S_k| > |S \cap S_k|$ then there is $j < k$ such that $|S \cap S_j| > |S' \cap S_j|$.*

The only difference between this definition and the original one [1, 16] is that there is no lexicographically preferred subbase if K is inconsistent.

Definition 11 *Let $KB = (K, S_1, \ldots, S_n)$ be a stratified knowledge base. A formula F is lexicographically entailed by KB, denoted $KB \vdash_{lex} F$, iff F is entailed by all lexicographically preferred subbases of KB.*

We now define the translation:

Definition 12 *Let T be a QCL base in normal form. The stratified knowledge base associated with T, denoted $Skb(T)$, is*

$$Skb(T) = (T^*, T_1, \ldots, T_n)$$

where $n = max\{k \mid F \in T, opt(F) = k\} - 1$,
T^* is obtained from T by replacing each occurrence of $\times$ by $\vee$, and
$T_i = \{A_1 \vee \ldots \vee A_i \mid 1 \leq i < k, A_1 \times \ldots \times A_k \in T\}$.

Proposition 8 *$T \mid\!\sim F$ iff $Skb(T) \vdash_{lex} F$.*

Note that in the definition of T_i the case $i = k$ is not needed since the corresponding disjunctions are already in T^*. It turns out that there is a second, equivalent translation. In Def. 12 we can equivalently define $T_i =$

$$\{\neg A_1 \wedge \ldots \wedge \neg A_{i-1} \wedge A_i \mid 1 \leq i < k, A_1 \times \ldots \times A_k \in T\}.$$

To illustrate the translation consider the hotel example given in the introduction (after $\neg hotel_1$ was learned). The stratified knowledge base associated with T is $Skb(T) = (T^*, T_1, T_2)$ where T^* consists of:

$\neg hotel_1$
$walking \vee hotel{-}transport \vee public{-}transport$
$hotel_1 \vee hotel_2 \vee hotel_3 \vee hotel_4$
$hotel_1 \rightarrow walking$
$hotel_2 \rightarrow \neg walking \wedge hotel{-}transport$
$hotel_3 \rightarrow \neg walking \wedge \neg hotel{-}transport \wedge public{-}transport$
$hotel_4 \rightarrow \neg walking \wedge \neg hotel{-}transport \wedge \neg public{-}transport$

and

$T_1 = \{walking\}$
$T_2 = \{walking \vee hotel{-}transport\}$

There is exactly one lexicographically preferred subbase, namely $T^* \cup T_2$. We thus have

$$Skb(T) \vdash_{lex} hotel_2$$

as intended.

3.3 Compilation to a classical KB

This step is described in [2]. The main idea behind the compilation is to replace lexicographic entailment from the original base by classical entailment from a compiled base, which contains either formulas from the original base or formulas subsumed by the original ones, obtained from the disjunction of some original formulas.

This idea has several implementations, namely Disjunctive Maxi-Adjustment (DMA), Iterative DMA and Whole-DMA.

- DMA replaces formulas involved in a conflict by disjunctions that restore consistency involving a minimum of these formulas (if any). Here the conflicting formulas are only computed once.

- Iterative DMA only adds pairwise disjunctions to the KB and iterates the detection of conflicting formulas in order to stay as close as possible to the original KB. Unfortunately, the approach is computationally costly since determining formulas involved in a conflict is on the second level of the polynomial hierarchy in complexity theory.

- In contrast, Whole-DMA does not detect formulas involved in conflicts, but works with disjunctions built from all the formulas. The advantage is that the complexity of the approach is on the first level of the polynomial hierarchy (SAT), and that in practice SAT solvers are now able to solve instances of the problem with several thousands of variables. The disadvantage is that the size of the compiled KB is likely to explode exponentially.

Note that for the hotel example discussed above the three approaches yield the classical knowledge base $T^* \cup \{walking \vee hotel{-}transport\}$.

For the details we refer to [2].

4 Applications

Qualitative choice logic has numerous useful applications.

- In design and configuration, intended properties can be described and ranked according to their desirability.

- In agent systems, intended actions to be performed by agents can be specified together with backup actions covering situations where the standard actions are inappropriate.

- In database and web applications, prioritized queries can be expressed which describe suboptimal results. For instance, one may start a query like: find a used convertible less than 2 years old in red, if red is unavailable then in black. This may be very useful for e-commerce applications.

Due to space limitations we will focus on configuration and prioritized queries in this paper.

The configuration task can informally be described as follows: given a set of components *Comp* determine a set $C \subseteq Comp$ such that C is a complete solution (no necessary component is lacking), contains no unnecessary elements, and satisfies certain requirements depending on the case at hand as much as possible. More precisely the following pieces of information are relevant:

- A description of the available components which can be chosen for a particular configuration or subconfiguration.

- A description of additional components which are necessary if some component/subconfiguration is chosen.

- A description of the properties of the components.

- A description of the particular case at hand.

- A description of the desired properties respectively intended components

Let's illustrate this using a trip planning example. The available main options are going by *plane*, by *train* or by *car*. If *plane* is chosen it is necessary to organize transportation to and from the airport. If *train* is chosen one has to get to the station and from the station to the final destination. For both cases taxis and buses are available. Here is a formalization of this information. It is convenient to use exclusive or ($\oplus$) in configuration examples.

$trip \leftrightarrow plane \oplus train \oplus car$
$\neg plane \vee \neg train \vee \neg car$
$train \leftrightarrow toSt$
$train \leftrightarrow fromSt$
$plane \leftrightarrow toAp$
$plane \leftrightarrow fromAp$
$toSt \leftrightarrow taxi{-}toSt \oplus bus{-}toSt$
$fromSt \leftrightarrow taxi{-}fromSt \oplus bus{-}fromSt$
$toAp \leftrightarrow taxi{-}toAp \oplus bus{-}toAp$
$fromAp \leftrightarrow taxi{-}fromAp \oplus bus{-}fromAp$

Note that we use $\leftrightarrow$ rather than $\rightarrow$ to describe the available alternatives. This is necessary to make sure that a component is contained in a configuration only if it is necessary.[4]

We next describe our preferences. For short trips we prefer *car* over *train*, for medium trips *train* over *car*. We never use the plane for such trips. Also, if we have heavy *luggage* we never take the train. For long distance trips our first preference is *plane*, followed by *train* and *car*. We also need to represent the background knowledge that a trip belongs to exactly one of the three categories:

$short \rightarrow car \times train$
$medium \rightarrow train \times car$
$long \rightarrow flight \times train \times car$
$luggage \rightarrow \neg train$
$short \oplus medium \oplus long$
$\neg short \vee \neg medium \vee \neg long$

The preferences for travelling to and from the airport respectively station are as follows:

$toSt \rightarrow taxi{-}toSt \times bus{-}toSt$
$fromSt \rightarrow taxi{-}fromSt \times bus{-}fromSt$
$toAp \rightarrow bus{-}toAp \times taxi{-}toAp$
$fromAp \rightarrow bus{-}fromAp \times taxi{-}fromAp$

Now given a description of the trip to be planned and the relevant requirements we obtain a suitable configuration. For instance, if in addition to the formulas above we have *trip*, *short* then the preferred model contains *car* and no other component. If we have *trip*, *short*, $\neg car$ we get $taxi{-}toSt, train, taxi{-}fromSt$. In each case we obtain a configuration which best satisfies our preferences.

The next example involves prioritized queries. Assume a database DB of certain items, say cars, with a description of their properties is given:

$car_1 \rightarrow BMW \wedge red \wedge \neg convertible$
$car_2 \rightarrow BMW \wedge green \wedge convertible$
$car_3 \rightarrow VW \wedge blue \wedge \neg convertible$
$car_4 \rightarrow VW \wedge blue \wedge convertible$

We assume DB also contains background knowledge of the kind

$$BMW \rightarrow \neg VW, red \rightarrow \neg blue, red \rightarrow \neg green$$

etc. together with information that exactly one car is to be chosen: $car_1 \vee car_2 \vee \ldots$ and the formulas $\{car_i \rightarrow \neg car_j \mid i < j\}$. Now a prioritized query is just a set of QCL formulas Q. An answer Ans is a disjunction of items such that $DB \cup Q \mid\!\sim Ans$. In the example we might have $Q = \{red \times blue, convertible \times \neg convertible\}$. This query would lead to the answer $car_1 \vee car_4$. car_1 is contained in the answer since it has the most preferred colour, car_4 since it is a convertible. If we add to Q the formula $BMW \times VW$ the answer becomes car_1.

Note that since the preference relation on models depends on the number of formulas satisfied to a particular degree we can give more weight to a particular criterion by adding a syntactic variant of a formula in Q. For instance, if we add $red \wedge \top \times blue$ to our original query then the colour criterion becomes more important and we obtain the answer BMW.

We conclude this section with a few remarks about the role of formulas in QCL theories. There is no syntactic distinction between formulas representing beliefs and formulas representing desires or intentions. For the applications discussed in this section this does not pose any problems. In contexts where the distinction is important it may be useful (and necessary to avoid wishful thinking, see [25]) to split the premises T into two subsets, a set K of classical formulas representing beliefs about the real world and a set D of QCL formulas representing preferences. A formula F then is believed to be true in the actual world if $K \vdash F$ and it is desired to be true if $K \cup D \mid\!\sim F$.

5 Alternative definitions of entailment

In Sect. 2 we defined entailment for QCL in terms of a lexicographic ordering on models, based on the number of formulas satisfied to a certain degree. This leads to an approach where solutions are preferred when they contain the highest number of most preferred options. For instance, if there are three choices with three options each, say

$$A_1 \times A_2 \times A_3, B_1 \times B_2 \times B_3, C_1 \times C_2 \times C_3$$

[4] The second formula is necessary because $plane \oplus train \oplus car$ is true also if all three alternatives are true.

then a model M_1 satisfying A_1, B_1 and C_3 is preferred over a model M_2 satisfying A_2, B_2 and C_1 because the number of formulas satisfied in M_1 with degree 1 is 2, the number in M_2 is 1.

This may not be wanted for all applications. In our example we might consider M_2 a reasonable alternative: although it gives us only 1 best choice, we still get two second best options. The following strengthening of the preference relation is based on subsets rather than the number of formulas satisfied with a particular degree

Definition 13 *Let $M^k(T)$ denote the set of formulas of T satisfied by a model M of T to degree k. A model M_1 of T is T-inclusion-preferred over a model M_2 if there is a k such that $M_1^k(T)$ is a strict superset of $M_2^k(T)$ and for all $j < k$: $M_1^j(T) = M_2^j(T)$. M is an inclusion-preferred model of T iff M is a maximally T-inclusion-preferred model of T.*

Note that inclusion preference implies the cardinality based preference introduced earlier but not vice versa. In the example above M_1 is not inclusion-preferred over M_2. We therefore get in general more maximally inclusion-preferred models and thus fewer conclusions, that is the inference relation $\mid\sim_{inc}$ (defined as $T \mid\sim_{inc} F$ iff F is true in all inclusion preferred models of T) is more cautious than $\mid\sim$.

It turns out that a result corresponding to Prop. 8 can be established for inclusion based preference using the same translation. The definition of an inclusion preferred subbase of a stratified knowledge base is obtained from Def. 10 by replacing clause 2. with

> if S' is a maximal consistent subset of $(K \cup S_1 \cup \ldots \cup S_n)$ containing K and for some $k \in \{1, \ldots, n\}$: $S \cap S_k$ is a proper subset of $S' \cap S_k$ then there is $j < k$ such that $S' \cap S_j$ is a proper subset of $S \cap S_j$.

We have the following proposition:

Proposition 9 *$T \mid\sim_{inc} F$ iff $Skb(T) \vdash_{inc} F$.*

Another possibility for defining preferred models is to just add up the (dis)satisfaction degrees of all formulas in T for each model and to prefer those models whose sum is minimal, that is whose overall dissatisfaction is minimal. In the example both M_1 and M_2 have overall degree 5 ($1 + 1 + 3$ and $2 + 2 + 1$, respectively), thus none of the models is preferred to the other.

Even more fine grained distinctions could be introduced by adding to each option in an ordered disjunction some integer (increasing in value from left to right) which could be used to compute the overall dissatisfaction degree of models.

A further study of these alternative notions of entailment is left for future work. A recent complexity result from Coste-Marquis and Marquis [9] may be relevant here: they prove that there is no way to compile any stratified knowledge base under inclusion preference in polynomial space whereas such a translation is possible under lexicographic preference by adding new propositional variables (the compiled base is "query equivalent" to the original one). This may be seen as an additional argument in favour of lexicographic preference.

6 Conclusions

We proposed in this paper a new nonmonotonic propositional logic for representing ranked options. The logic has a new connective called ordered disjunction. Since ordered disjunction is fully embedded in the language the ranking of the options may depend on the particular context, that is, it may depend on what else is true in the current situation.

There are numerous approaches to adding preferences to nonmonotonic logics and logic programs. For an overview of some of them see the discussion in [8] or the more recent [23]. Only few approaches allow for context dependent preferences. Existing work on context dependent preferences in nonmonotonic logics is based on explicit representations of a preference ordering together with names for default rules and sophisticated reformulations of the acceptable belief sets [5, 6] or makes heavy use of meta-predicates and compilation techniques [10, 12]. The availability of ordered disjunction in QCL allows context dependent preferences among properties of a problem solution to be expressed much more conveniently.

We investigated computational aspects of QCL and showed how sets of formulas can be translated to stratified knowledge bases and, by results in [2], to classical propositional knowledge bases. We indicated a number of potential applications of the logic, and we presented alternative definitions of the consequence relation for applications where the lexicographic ordering based on the number of best possible options is not adequate.

QCL is closely related to approaches in qualitative decision making [11]. Poole [21] aims at a combination of logic and decision theory. His approach incorporates quantitative utilities whereas our preferences are qualitative. Interestingly, Poole uses a logic *without* disjunction ("rather than using disjunction ... we want to use probability and decision theory to handle un-

certainty", Sect. 1.5) whereas we *enhance* disjunction.

In [4] a graphic representation, somewhat reminiscent of Bayes nets, for conditional preferences among feature values under the *ceteris paribus* principle is proposed, together with corresponding algorithms. We are able to represent preferences among arbitrary formulas and have developed a full fledged logic.

In a series of papers [14, 26, 15], originally motivated by [3], the authors propose viewing conditional desires as constraints on utility functions. Intuitively, $D(a|b)$ stands for: the b-worlds with highest utility satisfy a. Our interpretation of ranked options is very different. Rather than being based on decision theory our approach can be viewed as giving a particular interpretation to the ceteris paribus principle: a model M_1 is preferred over a model M_2 if there is a formula $F \in T$ satisfied to degree j by M_1 and to degree $k > j$ by M_2 provided M_1 satisfies the other formulas in T at least as well as M_2. The last phrase is made precise as follows: for each degree $i \leq j$ M_1 satisfies at least as many formulas in $T \setminus \{F\}$ as M_2 to degree i.

In future work we want to investigate combinations of *QCL* and existing product configuration methodologies, e.g. [24], extensions of the inference relation to non-classical formulas, and the first order case. An application of the ideas underlying *QCL* to answer set programming is described in [7].

Acknowledgements

We would like to thank Pierre Marquis, Ilkka Niemelä and Leon van der Torre for helpful comments. The third author was supported in part by the IUT de Lens, the Université d'Artois, the Nord/Pas-de-Calais Région under the TACT-TIC project and by the European Community FEDER Program.

Appendix: Proofs of propositions

This appendix contains proofs of Prop. 5, 6, 8 and 9. The proofs of Lemma 1, Lemma 2, and Prop. 1 follow immediately given Prop. 5 and 6 (namely after transformation to simple choice formulas). Proofs of Prop. 2 and 3 follow from Prop. 8 and the well-known properties of the lexicographic systems. Prop. 7 is a corrolary of Prop. 5 and 6. Prop. 4 is a corrolary of Prop. 7 and results in [2].

Fact 1 *Let F_1, F_2 be two arbitrary choice formulas. Let I be an interpretation, and $k \geq 1$ be such that $I \models_k F_1$. Then we also have: $I \models_k F_1 \times F_2$.*

Proof of Prop. 5 (ordered disjunction is associative)

First, it easy to check that the two formulas $((F_1 \times F_2) \times F_3)$ and $(F_1 \times (F_2 \times F_3))$ have the same optionality, namely:
$opt((F_1 \times F_2) \times F_3) = opt(F_1 \times (F_2 \times F_3)) = opt(F_1) + opt(F_2) + opt(F_3)$.

Let I be an interpretation, let us consider different cases of satisfaction of the formulas F_i by I:

- $I \models_k F_1$ then using Fact 1, we have $I \models_k F_1 \times F_2$, and again applying Fact 1, we get: $I \models_k (F_1 \times F_2) \times F_3$.
  On the other hand, using Fact 1, we also have: $I \models_k F_1 \times (F_2 \times F_3)$.

- $I \models_1 \neg F_1$ and $I \models_k F_2$. Then by definition, we have $I \models_{k+opt(F_1)} F_1 \times F_2$, and applying Fact 1, we get: $I \models_{k+opt(F_1)} (F_1 \times F_2) \times F_3$.
  On the other hand, using Fact 1, we also have: $I \models_k F_2 \times F_3$, and by definition $I \models_{k+opt(F_1)} F_1 \times (F_2 \times F_3)$.

- $I \models_1 \neg F_1$, $I \models_1 \neg F_2$ and $I \models_k F_3$. Then by definition, we have $I \models_1 \neg(F_1 \times F_2)$, and : $I \models_{k+opt(F_1)+opt(F_2)} (F_1 \times F_2) \times F_3$.
  On the other hand, we also have: $I \models_{k+opt(F_2)} F_2 \times F_3$, and $I \models_{k+opt(F_1)+opt(F_2)} F_1 \times (F_2 \times F_3)$.

- $I \models_1 \neg F_1$, $I \models_1 \neg F_2$ and $I \models_1 \neg F_3$. Then by definition, we have $I \models_1 \neg((F_1 \times F_2) \times F_3)$ and $I \models_1 \neg(F_1 \times (F_2 \times F_3))$.

Proof of Prop. 6
(strong equivalences of formulas without $\times$)

For the three equivalences, it is easy to check the equality of the optionality of the equivalent formulas.

1. We assume for the sake of simplicity, and without loss of generality, that $m \leq n$. The equivalence becomes:
   $(A_1 \times \ldots \times A_n) \vee (B_1 \times \ldots \times B_m) \doteq$
   $(A_1 \vee B_1) \times \ldots \times (A_n \vee B_n)$
   where for $i > m$, $B_i = \bot$ (since A_i is classically equivalent to $B_i \vee \bot$).
   Let us consider different cases of satisfaction of A_i's and B_j's by I.

   - there exists $i > 0$ and $j > 0$ such that $I \models \neg A_1 \wedge \ldots \wedge \neg A_{i-1} \wedge A_i$ and $I \models \neg B_1 \wedge \ldots \wedge \neg B_{j-1} \wedge B_j$.
     This implies that:
     - for $k < min(i, j)$, $I \models \neg(A_k \vee B_k)$, and for

$n \geq k \geq min(i,j) : I \models A_k \vee B_k$,
- $I \models_i A_1 \times \ldots \times A_n$
- $I \models_j B_1 \times \ldots \times B_m$

This leads by definition to:
$I \models_{min(i,j)} (A_1 \times \ldots \times A_n) \vee (B_1 \times \ldots \times B_m)$, and
$I \models_{min(i,j)} (A_1 \vee B_1) \times \ldots \times (A_n \vee B_n)$

- $\forall i \leq n, I \models \neg A_i$ and there exists $j > 0$ such that $I \models \neg B_1 \wedge \ldots \wedge \neg B_{j-1} \wedge B_j$. This implies that:
$I \models \neg(A_1 \vee B_1) \wedge \ldots \wedge \neg(A_{j-1} \vee B_{j-1}) \wedge (A_j \vee B_j) \wedge \ldots \wedge (A_n \vee B_n)$,
$I \models_1 \neg(A_1 \times \ldots \times A_n)$
$I \models_j B_1 \times \ldots \times B_m$

Hence by definition, we get:
$I \models_j (A_1 \times \ldots \times A_n) \vee (B_1 \times \ldots \times B_m)$, and
$I \models_j (A_1 \vee B_1) \times \ldots \times (A_n \vee B_n)$

- $\forall i \leq n, I \models \neg A_i$, and $\forall j \leq m, I \models \neg B_j$. This implies that: $\forall i \leq n, I \models \neg(A_i \vee B_i)$.
Hence: $I \models_1 \neg(A_1 \times \ldots \times A_n) \vee (B_1 \times \ldots \times B_m)$, and
$I \models_1 \neg((A_1 \vee B_1) \times \ldots \times (A_n \vee B_n))$.

2. Again, we assume for the sake of simplicity, and without loss of generality, that $m \leq n$. The equivalence becomes:

$(A_1 \times \ldots \times A_n) \wedge (B_1 \times \ldots \times B_m) \doteq (C_1 \times \ldots \times C_n)$
where
$C_i = [(A_1 \vee \ldots \vee A_i) \wedge B_i] \vee [A_i \wedge (B_1 \vee \ldots \vee B_i)]$,
and $B_i = \bot$ for $m < i$.

Let us consider different cases of satisfaction of A_i's and B_j's by I.

- there exists $i > 0$ and $j > 0$ such that $I \models \neg A_1 \wedge \ldots \wedge \neg A_{i-1} \wedge A_i$ and $I \models \neg B_1 \wedge \ldots \wedge \neg B_{j-1} \wedge B_j$.
This implies that:
- for $k < max(i,j)$, $I \models \neg C_i$, and for $n \geq k \geq max(i,j)$, $I \models C_k$,
- $I \models_i A_1 \times \ldots \times A_n$
- $I \models_j B_1 \times \ldots \times B_m$

This leads by definition to:
$I \models_{max(i,j)} (A_1 \times \ldots \times A_n) \wedge (B_1 \times \ldots \times B_m)$, and
$I \models_{max(i,j)} C_1 \times \ldots \times C_n$
- $\forall i \leq n, I \models \neg A_i$, or $\forall j \leq m, I \models \neg B_j$. This implies that: $\forall i \leq n, I \models \neg C_i$.Hence:
$I \models_1 \neg[(A_1 \times \ldots \times A_n) \wedge (B_1 \times \ldots \times B_m)]$, and $I \models_1 \neg[C_1 \times \ldots \times C_n]$.

3. $I \models \neg(A_1 \vee \ldots \vee A_n)$ iff $\forall 0 \leq i \leq n, I \models \neg A_i$ iff $I \models_1 \neg(A_1 \vee \ldots \vee A_n)$ iff $I \models_1 \neg(A_1 \times \ldots \times A_n)$.

Proof of Prop. 8 (sketch)
$T \mid\sim F$ iff $Skb(T) \vdash_{lex} F$

Let $Skb(T) = S = (T^*, T_1, \ldots, T_n)$ be the stratified base associated with T. In the following, M_1 and M_2 denote two models of T. $|M_1^p(T)|$ and $|M_2^p(T)|$ denote the number of simple choice logic formulas from T satisfied to a degree p by M_1 and M_2 respectively. In a similar way, $|M_1^p(S)|$ and $|M_2^p(S)|$ denote the number of propositional formulas from T_p satisfied by M_1 and M_2 respectively. The idea of the proof is to show that the lexicographic ordering between models of T which is based on the number of satisfied choice formulas from T is the same as the one based on the number of satisfied propositional formulas from $Skb(T)$.

Fact 2 $|M_1^1(S)| \geq |M_2^1(S)|$ *iff* $|M_1^1(T)| \geq |M_2^1(T)|$

Given, this fact, the proof of Proposition 8 follows immediately by applying iteratively the following lemma:

Lemma 3 *Let us assume that:*
$\forall j = 1, \ldots, i$, $|M_1^j(T)| = |M_2^j(T)|$ *iff* $|M_1^j(S)| = |M_2^j(S)|$.
Then:

- $|M_1^{i+1}(T)| = |M_2^{i+1}(T)|$ *if and only if* $|M_1^{i+1}(S)| = |M_2^{i+1}(S)|$,
and
- $|M_1^{i+1}(T)| > |M_2^{i+1}(T)|$ *if and only if* $|M_1^{i+1}(S)| > |M_2^{i+1}(S)|$,

Proof: Wk will only show the first item. The other case follows similarly by replacing the symbol = by >.

First note that:

$|M_1^i(S)| + |M_1^{i+1}(T)| =$
$|\{A_1 \times \ldots \times A_n \in T : M_1 \models A_1 \vee \ldots \vee A_i, and\ n \geq i+1\}|$
$+|\{A_1 \times \ldots \times A_n \in T : M_1 \models \neg A_1 \wedge \ldots \wedge \neg A_i \wedge A_{i+1},\ and\ n \geq i+1\}|$
$= |\{A_1 \times \ldots \times A_n \in T : M_1 \models A_1 \vee \ldots \vee A_i \vee A_{i+1}, and\ n \geq i+1\}|$

Hence:

(1) $|M_1^{i+1}(S)| = |M_1^i(S)| + |M_1^{i+1}(T)| - |size^{i+1}(T)|$
where : $size^{i+1}(T) = \{A_1 \times \ldots \times A_n \in T : n = i+1\}$

It is clear that any model M of T, satisfies $A_1 \vee \ldots \vee A_n$ where $A_1 \times \ldots \times A_n \in size^{i+1}(T)$.

Given (1) the proof follows straightforwardly. Indeed,

- if $|M_1^i(S)| = |M_2^i(S)|$ and $|M_1^{i+1}(T)| = |M_2^{i+1}(T)|$ then using (1) we also have:
  $|M_1^{i+1}(S)| = |M_2^{i+1}(S)|$

- if $|M_1^i(S)| = |M_2^i(S)|$ and $|M_1^{i+1}(S)| = |M_2^{i+1}(S)|$ then using again (1) we also have:
  $|M_1^{i+1}(T)| = |M_2^{i+1}(T)|$

Proof of Prop. 9 (sketch)
$T \vdash_{inc} F$ iff $Skb(T) \vdash_{inc} F$

In the following, $M_1^p(T)$ and $M_2^p(T)$ denote the set of simple choice logic formulas from T satisfied to a degree p by M_1 and M_2 respectively. In a similar way, $M_1^p(S)$ and $M_2^p(S)$ denote the set of propositional formulas from T_i satisfied by M_1 and M_2 respectively. We define Skb-inclusion-preference between models of T exactly like in Definition 13, by replacing $M_1^j(T)$ and $M_2^j(T)$ by $M_1^j(S)$ and $M_2^j(S)$ respectively. The idea of the proof is then to show that the T-inclusion-based ordering coincides with Skb-inclusion-based ordering. We first give two facts and show a lemma.

Fact 3 *If* $M \models_k A_1 \times \ldots \times A_n$ *then for any* $j = k, \ldots, n$ *we have :* $M \models A_1 \vee \ldots \vee A_j$.

Fact 4 *If* $M \models A_1 \vee \ldots \vee A_j$ *then there exists some* $k \leq j$ *such that :* $M \models_k A_1 \times \ldots \times A_n$

Lemma 4 *If there exists some k such that for all* $j = 1, \ldots, k$ *we have :* $M_1^j(T) = M_2^j(T)$ *then for all* $j = 1, \ldots, k$ *we have :* $M_1^j(S) = M_2^j(S)$. *The converse is also true.*

Proof :

- Assume that for all $j = 1, \ldots, k$ we have :
  $M_1^j(T) = M_2^j(T) \ldots\ldots (1)$
  and there exists some $i \leq k$ such that : $M_1^i(S) \neq M_2^i(S)$.
  Let $A_1 \vee \ldots \vee A_i$ be a formula in S_i satisfied by M_1 but falsfied by M_2. Using Fact 4, we have $M_1 \models_j A_1 \times \ldots \times A_n$ from some $j \leq i$. Using the hypthesis (1) this implies that $M_2 \models_j A_1 \times \ldots \times A_n$. Applying Fact 3, we get $M_2 \models A_1 \vee \ldots \vee A_i$ and hence a contradiction.

- The other sense follows the same schema. Namely, assume that for all $j = 1, \ldots, k$ we have :
  $M_1^j(S) = M_2^j(S) \ldots\ldots (2)$ and there exists some $i \leq k$ such that : $M_1^i(T) \neq M_2^i(T)$.

  Let $A_1 \times \ldots \times A_n$ be a formula in T satisfied by M_1 to a degree i but is not satisfied by M_2 to a degree i. Using Fact 3, we get $M_1 \models A_1 \vee \ldots \vee A_i$ (and by definition for k < i, $M_1 \not\models A_1 \wedge \ldots \wedge A_k$). Hypthesis (2) implies $M_2 \models A_1 \vee \ldots \vee A_i$. Applying Fact 4, we get $M_2 \models_k A_1 \times \ldots \times A_n$ for some $k \leq i$, hence a contradiction.

Given this lemma, we have.

Lemma 5 M_1 *is Skb-inclusion-preferred to* M_2 *iff* M_1 *is T-inclusion-preferred to* M_2.

Proof:

- Let M_1 be Skb-inclusion-preferred to M_2 but M_1 is not T-inclusion-preferred to M_2. By definition, M_1 is Skb-inclusion-preferred to M_2 implies that there exists some i such that for all $j < i$ we have $M_1^j(S) = M_2^j(S)$ and $M_2^i(S) \subset M_1^i(S)$. From Lemma 4, we also get : for all $j < i$ we have $M_1^j(T) = M_2^j(T)$.

  Moreover, $M_2^i(S) \subset M_1^i(S)$ means that there exists a formula $A_1 \vee \ldots \vee A_i$ which is satisfied by M_1 but not by M_2. This means that there exists a choice logic formula $A_1 \times \ldots \times A_n$ which is satisfied to degree i by M_1 but not by M_2 (which means that $M_2^i(T)$ is not included in $M_1^i(T)$).

  Now, assume that there exists a simple choice logic formula $A_1 \times \ldots \times A_n$ which is satisfied by M_2 to a degree i but not by M_1. From Fact 3, this implies that $M_2 \models A_1 \vee \ldots \vee A_i$, which implies $M_1 \models A_1 \vee \ldots \vee A_i$ (since $M_2^i(S) \subset M_1^i(S)$) and this contradicts the fact that $A_1 \times \ldots \times A_n$ is not satisfied by M_1 to a degree i.

- The proof is symmetric. Let M_1 be T-inclusion-preferred to M_2 but M_1 is not Skb-inclusion-preferred to M_2. By definition, M_1 is Incl-T-preferred to M_2 implies that there exists some i such that for all $j < i$ we have $M_1^j(T) = M_2^j(T)$ and $M_2^i(T) \subset M_1^i(T)$. From Lemma 4, we also get : for all $j < i$ we have $M_1^j(S) = M_2^j(S)$.

  Moreover, $M_2^i(T) \subset M_1^i(T)$ means that there exists a simple choice logic formula $A_1 \times \ldots \times A_n$ which is satisfied to a degree i by M_1 but not by M_2. This means that there exists a propositional formula $A_1 \vee \ldots \vee A_i$ in S_i which is satisfied by M_1 but not by M_2 (which means that $M_2^i(S)$ is not included in $M_1^i(S)$).

  Now, assume that there exists a formula $A_1 \vee \ldots \vee A_i$ in S_i which is satisfied by M_2 but not by M_1. Using Fact 4, this implies that $M_2 \models_i A_1 \times \ldots \times A_n$, which implies $M_1 \models_i A_1 \times \ldots \times A_n$ (since $M_2^i(T) \subset M_1^i(T)$) and this contradicts the fact that $A_1 \vee \ldots \vee A_i$ is not satisfied by M_1.

References

[1] Benferhat, S., Cayrol, C., Dubois, D., Lang, J., Prade, H., Inconsistency Management and Prioritized Syntax-Based Entailment, Proc. Intl. Joint Conference on Artificial Intelligence, Chambery, 1993, pp 640-645

[2] Benferhat, S., Kaci, S., Le Berre, D., Williams, M.-A., Weakening Conflicting Information for Iterated Revision and Knowledge Integration, Proc. 17th Intl. Joint Conference on Artificial Intelligence, Seattle, 2001, pp 109-115

[3] Boutilier, C., Towards a Logic for Qualitative Decision Theory, Proc. Principles of Knowledge Representation and Reasoning, KR-94, 1994, pp 75-86

[4] Boutilier, C.,Brafman, R.I., Hoos, H.H., Poole, D., Reasoning With Conditional Ceteris Paribus Preference Statements. Proc. UAI 99, 1999

[5] Brewka, G., Well-Founded Semantics for Extended Logic Programs with Dynamic Preferences, Journal of Artificial Intelligence Research, 4, 1996, pp 19-36

[6] Brewka, G., Representing Meta-Knowledge in Poole-Systems, Studia Logica 67, 2001, pp 153-165

[7] Brewka, G., Logic Programs with Ordered Disjunction, submitted for publication

[8] Brewka, G., Eiter, T., Preferred Answer Sets for Extended Logic Programs, Artificial Intelligence 109, 1999, pp 297-356

[9] Coste-Marquis, S. and Marquis, P., On Stratified Belief Base Compilations, submitted for publication

[10] Delgrande, J., Schaub, T., Tompits, H., Logic Programs with Compiled Preferences, Proc. European Conference on Artificial Intelligence, Berlin, 2000, pp 392-398

[11] Doyle, J., Thomason, R.H., Background to Qualitative Decision Theory, AI Magazine, Vol. 20, No. 2, Summer 1999, pp. 55-68

[12] Grosof, B., Diplomat: Compiling Prioritized Rules into Ordinary Logic Programs for E-commerce Applications, Proc. AAAI-99 (intelligent systems demonstration abstract), 1999

[13] Kraus, S., Lehmann, D., Magidor, M., Nonmonotonic Reasoning, Preferential Models and Cumulative Logics, Artificial Intelligence 44, 1990, pp 167-207

[14] Lang, J., Conditional Desires and Utilities - An Alternative Logical Approach to Qualitative Decision Theory, Proc. 12th European Conference on Artificial Intelligence, ECAI-96, 1996, pp 318-322

[15] Lang, J., van der Torre, L., Weydert, E., Utilitarian Desires, Autonomous Agents and Multi-Agent Systems, to appear

[16] Lehmann, D., Another perspective on default reasoning, Annals of Mathematics and Artificial Intelligence: 15(1), 1995, pp 61-82

[17] Lehmann, D., Magidor, M., What does a conditional knowledge base entail? Artificial Intelligence, 55, 1992, pp 1-60

[18] McCarthy, J., Circumscription – A Form of Non-monotonic Reasoning, Artificial Intelligence 13, 1980, pp 27-39

[19] McCarthy, J., Applications of Circumscription to Formalizing Common Sense Knowledge, Artificial Intelligence 28, 1986, pp 89-116

[20] Moore, R. C., Semantical Considerations on Nonmonotonic Logic, Artificial Intelligence 25, 1985, pp 75-94

[21] Poole, D., The Independent Choice Logic for Modelling Multiple Agents under Uncertainty, Artificial Intelligence 94(1-2), 1997, pp 7-56

[22] Reiter, R., A Logic for Default Reasoning, Artificial Intelligence 13, 1980, pp 81-132

[23] Schaub, T., Wang, K., A Comparative Study of Logic Programs with Preference, Proc. IJCAI-01, 2001

[24] Soininen, T., Niemelä, I., Tiihonen, J., Sulonen, R., Representing Configuration Knowledge With Weight Constraint Rules. Proceedings of the AAAI Spring Symposium on Answer Set Programming: Towards Efficient and Scalable Knowledge. Stanford, USA, 2001

[25] Thomason, R., Desires and Defaults: A Framework for Planning with Inferred Goals, Proc. Principles of Knowledge Representation and Reasoning, KR-2000, pp 702-713

[26] van der Torre, L., Weydert, E., Parameters for utilitarian desires in a qualitative decision theory. Applied Intelligence, 14, 2001, pp 285-301

Reducing Strong Equivalence of Logic Programs to Entailment in Classical Propositional Logic

Fangzhen Lin (flin@cs.ust.hk)
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Abstract

Recently Lifschitz, Pearce, and Valverde [2001] introduced a notion of strong equivalence between two logic programs, and showed that it can be captured in a 3-valued logic. In this paper, first for propositional logic programs with default negation, constraints, and disjunctions, we show that there is a simple mapping from these programs to propositional theories that reduces this notion of strong equivalence to entailment in classical propositional logic. Furthermore, we also provide a mapping in the other direction thus show that the problem of checking strong equivalence is co-NP-complete. We then consider logic programs with variables. One surprising result is that while the problem of deciding whether two logic programs are equivalent goes from decidable to undecidable when we move from logic programs without variables to ones with, the problem of deciding whether two logic programs are strongly equivalent remains to be co-NP-complete for logic programs with variables and constants.

1 Introduction

In this paper we shall consider logic programs with rules of the following form:

$$h_1; \cdots; h_k \leftarrow l_1, \cdots, l_n$$

where h_i's are atoms, and l_i's are either atoms or literals of the form $\mathsf{not}\, p$ for some atom p. So a logic program here can have default negation (not), constraints (when $k = 0$), and disjunctions in the head of its rules. The semantics of these programs are given by answer sets as defined in [Gelfond and Lifschitz, 1991].

Two such logic programs P_1 and P_2 are said to be *equivalent* if they have the same answer sets, and *strongly equivalent* [Lifschitz *et al.*, 2001] if for any logic program P, $P \cup P_1$ and $P \cup P_2$ are equivalent.

Questions regarding whether two logic programs are strongly equivalent are interesting for a variety of reasons. For instance, Lifschitz, Pearce, and Valverde [2001] argued that in order to see whether a set of rules can always be replaced by another one regardless of the context, one should check whether the two sets of rules are strongly equivalent. For instance, the set of single rule $p \leftarrow p$ is strongly equivalent to the empty set, so this rule can always be eliminated from any logic program. However, the set $\{p \leftarrow q.\ \ q \leftarrow p\}$ is equivalent, but not strongly equivalent to the empty set, so the pair of rules cannot be eliminated regardless of the context: in the presence of q the first rule can be used to derive p. As another example, while the rule $p \leftarrow \mathsf{not}\, p$ cannot in general be eliminated from a logic program, it can if the program contains the two rules $p \leftarrow \mathsf{not}\, q$ and $p \leftarrow\ q$: $\{p \leftarrow \mathsf{not}\, q.\ p \leftarrow\ q.\}$ and $\{p \leftarrow \mathsf{not}\, q.\ p \leftarrow\ q.\ p \leftarrow \mathsf{not}\, p.\}$ are strongly equivalent.

It is clear from the definition that strong equivalence implies equivalence. As it turns out, strong equivalence happens to have much better computational properties. So one can use the former to approximate the latter in applications such as program and query optimizations where ideally the latter should be used. In fact, that was exactly what Sagiv [1988] did for datalog programs: he proposed a notion of uniform equivalence and showed that in contrast to equivalence which is not decidable, uniform equivalence, which is the same as strong equivalence as we shall see later, is tractable. He then proposed an optimization algorithm based on uniform equivalence for eliminating redundant rules and redundant atoms in the body of a rule. For instance, the atom $f(w, y)$ in the body of the rule $g(x, y, z) \leftarrow\ g(x, w, z), f(w, y), f(w, z), f(z, z), f(z, y)$ can be eliminated since the rule is uniformly equivalent to $g(x, y, z) \leftarrow g(x, w, z), f(w, z), f(z, z), f(z, y)$.

Lifschitz, Pearce, and Valverde [2001] showed that checking for strong equivalence between two logic programs can be done in the logic of here-and-there, a three-valued non-classical logic somewhere between classical logic and intuitionistic logic. More recently, Turner [2001a] gave a model-theoretic characterization of strong equivalence in terms of pairs of sets of atoms. In this paper we shall provide a simple mapping from logic programs to propositional theories that reduces strong equivalence to entailment in classical propositional logic. Our mapping is motivated by both Turner's model-theoretic characterization and the translation of logic

programs to circumscriptive propositional theories in [Lin, 1991], which was in turn derived from the mapping from default logic to Lin and Shoham's logic of knowledge and justified assumption in [Lin and Shoham, 1992], a nonmonotonic bi-modal logic.

This paper is organized as follows. We shall first consider propositional logic programs. In section 2, we shall give a translation from logic programs to propositional theories that reduces strong equivalence of logic programs to entailment in propositional logic. In section 3, we shall give a converse translation from clauses to rules, thus showing that strong equivalence is co-NP-complete. In section 4, we consider logic programs that may contain variables and constants but not proper functions, i.e. those with arities greater than 0, and show that the problem of checking the strong equivalence of two logic programs with variables remains to be co-NP-complete. In section 5 we consider a more general notion of equivalence, and discuss some related work. Finally in section 6 we conclude this paper.

2 From strong equivalence to propositional entailment

In this section, we assume a fixed propositional language L. We assume that L has two special members: *true*, a tautology, and *false*, negation of a tautology. All logic programs in this section are supposed to be written in this language, i.e. all atoms in the logic programs come from L. For each atom $p \in L$, we assume that p' is a new atom not in L.

As mentioned above, a logic program P is a finite set of rules of the following form:

$$h_1; \cdots; h_k \leftarrow p_1, \cdots, p_m, \mathsf{not}\, p_{m+1}, \cdots, \mathsf{not}\, p_n. \quad (1)$$

where $h_1, ..., h_k, p_1, ..., p_n$ are atoms in L. Now for each rule (1) in P, let $\Gamma(P)$ contain the following two sentences:

$$p_1 \wedge \cdots \wedge p_m \wedge \neg p'_{m+1} \wedge \cdots \wedge \neg p'_n \supset h_1 \vee \cdots \vee h_k, \quad (2)$$

$$p'_1 \wedge \cdots \wedge p'_m \wedge \neg p'_{m+1} \wedge \cdots \wedge \neg p'_n \supset h'_1 \vee \cdots \vee h'_k. \quad (3)$$

Notice that if $m = n = 0$, then the left sides of the implications in (2) and (3) are considered to be *true*, and if $k = 0$, then the right sides of the implications in (2) and (3) are considered to be *false*.

As it turns out, this translation captures strong equivalence in propositional logic under the assumption that $p \supset p'$ for each p in the language:[1]

Theorem 1 *For any two logic programs P_1 and P_2, they are strongly equivalent iff the following two assertions hold:*

$$\{p \supset p' \mid p \in L\} \cup \Gamma(P_1) \models \Gamma(P_2),$$
$$\{p \supset p' \mid p \in L\} \cup \Gamma(P_2) \models \Gamma(P_1),$$

where "$\models$" is logical entailment in propositional logic.

[1] Independently, [Pearce *et al.*, 2001] proposed a similar translation for propositional logic programs with nested expressions but without disjunctions.

Proof: The proof of this theorem is based on Turner's result in [Turner, 2001a]. Let $X \subseteq Y$ be two sets of atoms. According to Turner, the pair (X, Y) is called an HT-model if both X and Y are closed under P^Y, where

- the program P^Y, the reduct of P on Y, is obtained from P as follows: for any rule $p_1; \cdots; p_k \leftarrow G$ in P, if G has a literal $\mathsf{not}\, q$ such that $q \in Y$, then delete this rule; otherwise, delete all the literals of the form $\mathsf{not}\, q$ in G; and
- a set of literals S is closed under a program Q without negation if for any rule $p_1; \cdots; p_k \leftarrow G$ in Q, if all the atoms in G are in S, then for some $1 \leq i \leq n$, p_i is in S.

Turner [2001a] showed that for any two programs P_1 and P_2, they are strongly equivalent iff they have the same HT-models. So to prove the theorem, we show that there is a one-to-one correspondence between the HT-models of P and the pairs: $(M_L, M_{L'})$, where M is a model of $\Sigma \cup \Gamma(P)$, and

$$\Sigma = \{p \supset p' \mid p \in L\},$$
$$M_L = \{p \mid p \in L \text{ and } M \models p\},$$
$$M_{L'} = \{p \mid p \in L \text{ and } M \models p'\}.$$

If M is a model of $\Sigma \cup \Gamma(P)$, then it can be shown that $(M_L, M_{L'})$ is an HT-model of P:

- $M_L \subseteq M_{L'}$ because $M \models \Sigma$.
- M_L is closed under $P^{M_{L'}}$ because for each rule (1) in P, M satisfies (2).
- $M_{L'}$ is closed under $P^{M_{L'}}$ because for each rule (1) in P, M satisfies (3).

Conversely, if (X, Y) is an HT-model of P, then we can construct the following truth assignment M: $M \models p$ iff $p \in X$, and $M \models p'$ iff $p \in Y$. Then it can be shown that M is a model of $\Sigma \cup \Gamma(P)$. ■

Examples

1. Let $P_1 = \{p \leftarrow p\}$. $\Gamma(P_1)$ is $\{p \supset p, p' \supset p'\}$, which is equivalent to tautology. So P_1 is strongly equivalent to the empty set, and can always be deleted from a logic program.
2. Let $P_2 = \{p \leftarrow \mathsf{not}\, p\}$. $\Gamma(P_2) = \{\neg p' \supset p, \neg p' \supset p'\}$, which is equivalent to $\{p'\}$. So P_2 is not strongly equivalent to the empty set, thus cannot in general be deleted.
3. Consider $P_3 = \{p \leftarrow q.\ p \leftarrow \mathsf{not}\, q\}$. $P_2 \cup P_3$ is strongly equivalent to P_3, thus in the presence of the rules in P_3, the rule in P_2 can always be deleted: $\Gamma(P_3)$ is $\{q \supset p, q' \supset p', \neg q' \supset p, \neg q' \supset p'\}$, which is equivalent to $\{q \supset p, \neg q' \supset p, p'\}$. Thus $\Gamma(P_3)$ entails $\Gamma(P_2)$.
4. P_3 and $P_4 = \{p\}$ are equivalent but not strongly equivalent. However, $P_3 \cup \{\leftarrow q\}$ and $P_4 \cup \{\leftarrow q\}$ are strongly equivalent: this is easy to check as $\Gamma(\{\leftarrow q\})$ is equivalent to $\neg q \wedge \neg q'$. This means the two rules in P_3 can always be replaced by the single rule in P_4 under the constraint $\leftarrow q$.

5. $P_5 = \{q \leftarrow p\}$ and $P_6 = \{q \leftarrow p. \quad \leftarrow p, \text{not } q\}$ are strongly equivalent: $\Gamma(P_5)$ is equivalent to $(p \supset q) \wedge (p' \supset q')$, and $\Gamma(P_6)$ to $(p \supset q) \wedge (p' \supset q') \wedge (p \supset q') \wedge (p' \supset q')$; while these two sentences are not logically equivalent, they are under the axiom $q \supset q'$. This example shows that the set of constraints $\{p \supset p' \mid p \in L\}$ is necessary in Theorem 1.
6. For an example of disjunctive logic programs, consider the following two logic programs which have shown to be strongly equivalent in [Lifschitz *et al.*, 2001]: $P_7 = \{p; q. \quad \leftarrow p, q\}$ and $P_8 = \{p \leftarrow \text{not } q. \; q \leftarrow \text{not } p. \quad \leftarrow p, q\}$. $\Gamma(P_7)$ is $\{p \vee q, p' \vee q', \neg(p \wedge q), \neg(p' \wedge q')\}$, and $\Gamma(P_8)$ is

$$\{\neg q' \supset p, \neg q' \supset p', \neg p' \supset q, \neg p' \supset q', \neg(p \wedge q), \neg(p' \wedge q')\},$$

which is equivalent to $\{q' \vee p, q' \vee p', p' \vee q, \neg p \vee \neg q, \neg p' \vee \neg q'\}$. It can be easily verified by resolution that $\Gamma(P_7)$ and $\Gamma(P_8)$ are equivalent under $\{p \supset p', q \supset q'\}$.

3 From propositional entailment to strong equivalence

We now show that given two sets of clauses, the problem of checking whether one of them entails the other can be reduced to that of checking whether two logic programs are strongly equivalent. Together with the result in the previous section, this then shows that the problem of checking whether two logic programs are strongly equivalent is co-NP-complete.

Let L be our language. For each $p \in L$, let $\hat{p}$ be a new proposition. In the following, we let $\hat{L} = L \cup \{\hat{p} \mid p \in L\}$, and denote by Π the set of following rules:[2] for each $p \in L$,

$$p \leftarrow \text{not } \hat{p},$$
$$\hat{p} \leftarrow \text{not } p,$$
$$\leftarrow p, \hat{p}.$$

This is by now a well-known technique of making atom p an assumption, i.e. one can assume either p or its negation $\hat{p}$, but not both.

For any clause $c = p_1 \vee \cdots \vee p_m \vee \neg p_{m+1} \vee \cdots \vee \neg p_n$ in L, let $\Delta(c)$ be the following constraint in $\hat{L}$:

$$\leftarrow \hat{p}_1, ..., \hat{p}_m, p_{m+1}, ..., p_n.$$

For any set S of clauses in L, let $\Delta(S) = \{\Delta(c) \mid c \in S\}$.

Theorem 2 *Let S_1 and S_2 be two sets of propositional clauses in L. $S_1 \models S_2$ iff $\Pi \cup \Delta(S_1)$ and $\Pi \cup \Delta(S_1) \cup \Delta(S_2)$ are strongly equivalent in $\hat{L}$.*

Proof: It is easy to see that for any set S of clauses in L, and any program P in $\hat{L}$, a set of atoms $M \subseteq \hat{L}$ is a stable model of $\Pi \cup \Delta(S) \cup P$ iff M contains exactly one of p and $\hat{p}$ for any $p \in L$, and satisfies S and P (as a set of clauses). From this and the definition of strong equivalence, the theorem follows. ■

From Theorem 1 and Theorem 2, we have the following corollary:[3]

Corollary 2.1 *The problem of checking the strong equivalence of two logic programs, with or without disjunctions, is co-NP-complete.*

Notice that we made use of constraints in our translation above from clauses to logic programs. In general, constraints cannot be replaced by rules. For instance, we can show that the constraint $\leftarrow p$ is not strongly equivalent to any set of rules with non-empty heads.

Proposition 3.1 *There does not exist any set P of rules with non-empty heads such that P and $\{\leftarrow p\}$ is strongly equivalent.*

Proof: Suppose P is such a program. Then $\Gamma(P)$ is a set of sentences of the form $A \supset \alpha$, where α is a disjunction of atoms. Now let M be an interpretation that assigns every proposition true, then M satisfies $\Gamma(P)$ and $\{q \supset q' \mid q \in L\}$, but does not satisfy $\Gamma(\{\leftarrow p\}) = \{\neg p, \neg p'\}$. So by Theorem 1, P and $\{\leftarrow p\}$ cannot be strongly equivalent. ■

This result may seem counter-intuitive as there is a well-known translation from constraints to rules using a special "fail" atom: let P be any program, and P' the result of replacing each constraint $\leftarrow Body$ in P by $fail \leftarrow Body$ and adding the new rule:

$$dummy \leftarrow \text{not } dummy, fail$$

where $fail$ and $dummy$ are two new atoms. Clearly, P and P' are equivalent. In fact, for any program Q that does not contain $fail$ and $dummy$, $P \cup Q$ and $P' \cup Q$ are equivalent. However, it is obvious that they are not strongly equivalent.

Turner[4] proposed the following translation from clauses to normal logic programs without using constraints. Again for each $p \in L$, let $\hat{p}$ be a new proposition. Let $\hat{L} = L \cup \{\hat{p} \mid p \in L\}$. Now denote by Π' the set of following rules: for each $p \in L$,

$$p \leftarrow \text{not } \hat{p}, \text{not } fail$$
$$\hat{p} \leftarrow \text{not } p, \text{not } fail$$
$$fail \leftarrow p, \hat{p}, \text{not } fail,$$

where $fail$ is a new proposition.

Now let h^+ and h^- be the following mappings: for each $p \in L$, $h^+(p) = p$ and $h^+(\neg p) = \hat{p}$, and $h^-(p) = \hat{p}$ and $h^-(\neg p) = p$. For any clause $c = l_1 \vee \cdots \vee l_m$, $m \geq 1$, in L, let $\Delta'(c)$ be the following rule in $\hat{L}$:

$$h^+(l_1) \leftarrow h^-(l_2), ..., h^-(l_m).$$

[2] As we have seen in the last section, we could replace the first two rules by the disjunction $p; \hat{p}$.

[3] Turner [2001b] independently showed that the problem of checking whether two logic programs are strongly equivalent is in co-NP.

[4] Personal communication, 2001.

For a set S of clauses in L, let $\Delta'(S) = \{\Delta'(c) \mid c \in S\}$. Turner showed that Theorem 2 holds for this translation: for any two sets of clauses S_1 and S_2, $S_1 \models S_2$ iff $\Pi' \cup \Delta'(S_1)$ and $\Pi' \cup \Delta'(S_1) \cup \Delta'(S_2)$ are strongly equivalent. Thus, even for normal logic programs, checking whether two programs are strongly equivalent is co-NP-complete.

4 Logic programs with variables but without functions

For most applications of logic programming, rules normally have variables. In this section we try to extend our results to these rules.

In the following, we shall assume a first-order language L that has a finite set of predicates, a finite set of constants, and no other functions. Unless stated otherwise, in the following, a logic program is a finite set of rules in L. Let P be such a program. A *domain* for P is then a finite set of elements that include all constants in P. Given a domain D for P, the instantiation of P on D, written P_D, is the set of rules resulted from substituting elements of D for the variables in the rules of P.

We say that two logic programs P and Q are (resp. strongly) equivalent if for any domain D of P and Q, P_D and Q_D are (resp. strongly) equivalent as two propositional logic programs. Notice that a domain for both P and Q must contain all constants in $P \cup Q$.

Unfortunately, the problem of checking whether two programs with variables are equivalent is undecidable. This can be seen from a similar result about *datalog programs* in deductive databases.

A datalog program is a set of *definite rules* that does not have any constants and function symbols, where a rule is definite if it has no negation and disjunction. A relation in a datalog program is called *extensional* if there is no rule about it in the program; otherwise, it is called *intensional*. In deductive database literature, two datalog programs are said to be equivalent if for any given finite extensional relations, the two programs yield the same intensional relations (cf. [Sagiv, 1988]). It is well-known in deductive databases that the problem of checking if two datalog programs are equivalent is undecidable [Shmueli, 1986]. From this, we conclude that the problem of checking whether two logic programs with variables are equivalent is also undecidable because of the following simple observation. Let P and Q be two datalog programs such that $E_1, ..., E_k$ are their extensional relations. Let $\hat{E}_1, ..., \hat{E}_k$ be new relations of the same arities with $E_1, ..., E_k$, respectively. Let Σ be the set of following rules:

$$E_i(\vec{x}_i) \leftarrow \text{not } \hat{E}_i(\vec{x}_i),$$
$$\hat{E}_i(\vec{x}_i) \leftarrow \text{not } E_i(\vec{x}_i),$$

for each $1 \leq i \leq k$. Clearly, P and Q are equivalent as two datalog programs iff $P \cup \Sigma$ and $Q \cup \Sigma$ are equivalent according to our definition.

In deductive databases, Sagiv [1988] and Maher [1988] also introduced a stronger notion of equivalence between two datalog programs: two datalog programs are *uniformly equivalent* if given any finite extensional relations, and any initial finite intensional relations, the two programs always yield the same intensional relation. It can be shown that two datalog programs are uniformly equivalent iff they are strongly equivalent according to our definition above. In contrast to equivalence, Sagiv [1988] showed that checking whether two datalog programs are uniformly equivalent is a tractable problem and provided an algorithm for doing so. He also showed that using the algorithm one can sometimes improve the efficiency of a datalog program by eliminating redundant subgoals in the body of a rule.

The class of logic programs considered in this paper is much more expressive than that of datalog programs. Even in the propositional case, checking strong equivalence is not a tractable problem. Still, unlike equivalence, for which there is no algorithm to verify it when there are variables in the rules, we shall show that the problem of verifying strong equivalence for programs with variables continues to be decidable, and in fact belongs to the same complexity class as that for propositional logic programs.

To this end, we first generalize Theorem 1 to programs with variables. To do this, we need to introduce some notations. In the following, we use $\models_{FM}$ to denote entailment under finite models. That is, for any theory T and sentence φ, $T \models_{FM} \varphi$ if for every first-order structure M with a finite domain, M satisfies φ whenever it satisfies every sentence in T. The meaning of $T \models_{FM} T'$ is defined similarly when both T and T' are sets of sentences.

As in the propositional case, for each predicate p in the language L, let p' be a new predicate of the same arity as p. We also need an axiom for each predicate p saying that p implies p'. In the following, we let

$$\Sigma = \{(\forall \xi).p(\xi) \supset p'(\xi) \mid p \text{ is a predicate in } L\}.$$

Now for each rule

$$h_1(u_1); \cdots; h_k(u_k) \leftarrow p_1(v_1), \cdots, p_m(v_m), \text{not } p_{m+1}(v_{m+1}), \cdots, \text{not } p_n(v_n)$$

in P, let $\Gamma(P)$ contain the following two sentences:

$$(\forall \xi).[p_1(v_1) \wedge \cdots \wedge p_m(v_m) \wedge \neg p'_{m+1}(v_{m+1}) \wedge \cdots \wedge \neg p'_n(v_n)] \supset h_1(u_1) \vee \cdots \vee h_k(u_k),$$
$$(\forall \xi).[p'_1(v_1) \wedge \cdots \wedge p'_m(v_m) \wedge \neg p'_{m+1}(v_{m+1}) \wedge \cdots \wedge \neg p'_n(v_n)] \supset h'_1(u_1) \vee \cdots \vee h'_k(u_k),$$

where ξ is the tuple of variables in the rule.

From Theorem 1, we immediately have the following results:

Proposition 4.1 *For any domain D of P and Q, P_D and Q_D are strongly equivalent iff for any interpretation I with domain D, if I satisfies $Uni \cup \Sigma$, then I satisfies*

$\Gamma(P)$ *iff it satisfies* $\Gamma(Q)$, *where* Uni *is the set of unique names axioms about the constants in* P *and* Q*:*

$$Uni = \{c \neq d \mid \text{for any pair of distinct constants } c \text{ and } d \text{ in } P \text{ and } Q\}.$$

Notice that the unique names axioms are implicitly assumed in logic programs.

From this proposition, we have:

Theorem 3 *Two logic programs* P *and* Q *are strongly equivalent iff*

$$Uni \cup \Sigma \cup \Gamma(P) \models_{FM} \Gamma(Q),$$
$$Uni \cup \Sigma \cup \Gamma(Q) \models_{FM} \Gamma(P),$$

where Uni *is as given in Proposition 4.1 above.*

In general, entailment under finite models is not decidable - it follows from Trahtenbrot's theorem (cf. [Ebbinghaus and Flum, 1999]) that the class of the sentences true in all finite models is not even recursively enumerable. However, for the class of sentences in Theorem 3, it is decidable.

To show this, we need the following lemma from first-order logic.

Lemma 4.1 *Let* φ *be a formula of form*

$$\forall x_1 \cdots \forall x_n \exists y_1 \cdots \exists y_m . \phi,$$

where $n \geq 1$, $m \geq 0$, *and* ϕ *is a formula that contains no quantifiers, constants and function symbols. Then for any* $k \geq n$, φ *is valid iff it is true for all interpretations with a domain of* k *objects.*

Proof: Exercise 2.58, page 74, of [Mendelson, 1987]. ■

For each rule, we call the number of variables in it its *variable rank*. The variable rank of a program is then the maximum of the variable ranks of rules in the program.

Theorem 4 *Let* P *and* Q *be two logic programs. Let* n *be the maximum of the variable ranks of* P *and* Q, *and* m *the number of constants in* $P \cup Q$. *Then* P *and* Q *are strongly equivalent iff* P_D *and* Q_D *are strongly equivalent for some domain* D *of* $m + n$ *elements for* P *and* Q.

Proof: Notice that for any program R, $\Gamma(R)$ is a finite set of sentences of the form: $\forall x_1 \cdots \forall x_k . W$, where W is a formula that contains no function symbols of arities greater than 0. So $\Gamma(R)$ is equivalent to a sentence also of such a form such that k is the variable rank of R. Without loss of generality, let the variable ranks of P and Q be n and t, respectively, and $n \geq t$, and that $\Gamma(P)$ and $\Gamma(Q)$ are equivalent to $\forall x_1 \cdots \forall x_n . W$ and $\forall y_1 \cdots \forall y_t . W'$, respectively.

It is easy to see that Σ is equivalent to a sentence of the form $(\forall \vec{z}).A(\vec{z})$ such that A does not contain any quantifiers, constants, and functions.

The unique names axioms Uni about constants in these two programs can be simulated by introducing m new unary predicates: for instance, the unique names axioms about three constants c_1, c_2, and c_3 can be axiomatized by the following axiom:

$$U_1(c_1) \wedge \neg U_1(c_2) \wedge \neg U_1(c_3) \wedge$$
$$U_2(c_2) \wedge \neg U_2(c_1) \wedge \neg U_2(c_3) \wedge$$
$$U_3(c_3) \wedge \neg U_3(c_1) \wedge \neg U_3(c_2).$$

In the following, we shall understand Uni to be an axiom of such nature.

Now assuming that $\vec{x}$, $\vec{y}$, and $\vec{z}$ have no common elements, then by Theorem 3, P and Q are strongly equivalent iff

$$\models_{FM} [Uni \wedge (\forall \vec{z})A \wedge (\forall \vec{x})W] \supset (\forall \vec{y})W',$$
$$\models_{FM} [Uni \wedge (\forall \vec{z})A \wedge (\forall \vec{y})W'] \supset (\forall \vec{x})W,$$

iff

$$\models_{FM} (\forall \vec{y})(\exists \vec{x})(\exists \vec{z}).\neg Uni \vee \neg A \vee \neg W \vee W',$$
$$\models_{FM} (\forall \vec{x})(\exists \vec{y})(\exists \vec{z}).\neg Uni \vee \neg A \vee \neg W' \vee W,$$

iff

$$\models_{FM} (\forall \vec{u})(\forall \vec{y})(\exists \vec{x})(\exists \vec{z}).\neg Uni(\vec{c}/\vec{u}) \vee \neg A \vee \neg W(\vec{c}/\vec{u}) \vee W'(\vec{c}/\vec{u}),$$
$$\models_{FM} (\forall \vec{u})(\forall \vec{x})(\exists \vec{y})(\exists \vec{z}).\neg Uni(\vec{c}/\vec{u}) \vee \neg A \vee \neg W'(\vec{c}/\vec{u}) \vee W(\vec{c}/\vec{u}),$$

where $\vec{c}$ is the tuple of m constants in Uni, $\vec{u}$ a tuple of m new variables, and for any formula φ, $\varphi(\vec{c}/\vec{u})$ is the result obtained from φ by replacing each constant in $\vec{c}$ by its corresponding variable in $\vec{u}$.

Thus by Lemma 4.1, P and Q are strongly equivalent iff the following two conditions hold:

1. $(\forall \vec{u})(\forall \vec{y})(\exists \vec{x})(\exists \vec{z}).\neg Uni(\vec{c}/\vec{u}) \vee \neg A \vee \neg W(\vec{c}/\vec{u}) \vee W'(\vec{c}/\vec{u})$ is true in every interpretation of size $m+n$.
2. $(\forall \vec{u})(\forall \vec{x})(\exists \vec{y})(\exists \vec{z}).\neg Uni(\vec{c}/\vec{u}) \vee \neg A \vee \neg W'(\vec{c}/\vec{u}) \vee W(\vec{c}/\vec{u})$ is true in every interpretation of size $m+n$.

The above two conditions hold iff for some domain D of $m + n$ objects, P_D and Q_D are equivalent as two propositional logic programs. ■

Corollary 4.1 *The question of whether any two logic programs that may contain variables and constants but not proper functions are strongly equivalent is co-NP-complete.*

Proof: From Theorem 5 and the fact that the size of P_D is polynomial in terms of the size of P and D. ■

It is unlikely that this result will hold for logic programs with function symbols as these programs cannot be instantiated into finite propositional ones.

5 Remarks

5.1 Relative equivalence

In the most general case, we can define equivalence between two logic programs relative to a set of atoms: two

logic programs P_1 and P_2 are said to be *equivalent with respect to a set* A of atoms if for any program P that mention only atoms in A, $P_1 \cup P$ and $P_2 \cup P$ are equivalent. Clearly, two programs are equivalent if they are equivalent w.r.t. the empty set, and strongly equivalent if they are equivalent w.r.t. the set of all atoms in the language.

It is possible that different applications may call for different notions of equivalence between logic programs. For the purpose of logic program optimization, perhaps the most appropriate is the one like that for datalog programs. For instance, the following program [Niemelä, 1999] for solving 3-color graph coloring problem is typical of those in answer set programming applications:

```
color(red).
color(blue).
color(yellow).
col(X,red) :- node(X), not col(X, blue),
              not col(X,yellow).
col(X,blue) :- node(X), not col(X, red),
               not col(X,yellow).
col(X,yellow) :- node(X), not col(X, blue),
                 not col(X,red).
fail :- edge(X,Y), color(C), col(X,C), col(Y,C).
```

Notice here that there is no rules about `edge` and `node` relations - these are supposed to be extensional predicates whose definitions will be provided once a specific graph is given. If one wants to optimize this program by finding an equivalent one that can be more efficiently computed using, say *smodels*, then the equivalence should be proved w.r.t. extensional predicates. However, as we know that this notion of equivalence is not in general decidable, strong equivalence may well turn out to be a useful approximation.

Another possible use of this notion of relative equivalence is to study some program transformation rules that are not, but almost, strongly equivalent. For instance, the transformation given in Section 3 for eliminating constraints are equivalent w.r.t. any language that does not contain *fail* and *dummy*.

5.2 Related work

This work is closely related to and strongly influenced by [Lifschitz *et al.*, 2001; Turner, 2001a; 2001b]. Semantically, as can be seen from the proof of Theorem 1, the ordinary propositions correspond to the first element, and the primed the second element in Turner's HT-model. Similarly, the ordinary propositions are "here" and primed "there" in the logic of here-and-there. Indeed [Pearce *et al.*, 2001] proposed a translation of logic of here-and-there to propositonal logic using similar techniques.

There are some differences in the classes of logic programs considered here and in [Lifschitz *et al.*, 2001; Turner, 2001a; 2001b]. On the one hand, in the propositional case, the class of logic programs considered in [Lifschitz *et al.*, 2001; Turner, 2001a; 2001b] is much more expressive than that considered in this paper. Their logic programs, called nested formulas,[5] allow rules of the form $\varphi \leftarrow \phi$, where φ and ϕ are formulas constructed from atoms using "not", ";", and ",". For instance, $(\text{not } a); (\text{not } b, a) \leftarrow (\text{not } c), ((\text{not } a); d)$ is a rule. While we have not proved it, we believe our translation can be extended to rules of this form as well: again each rule will be translated to two formulas: in one of them each atom is replaced by a new one formed by appending prime to its end, and in the other only atoms under the scope of not are so changed; and in both formulas, "not" will be replaced by $\neg$, ";" by $\vee$, and "," by $\wedge$. For instance, the above rule will be translated into the following two formulas:

$$\neg c' \wedge (\neg a' \vee d) \supset (\neg a' \vee (\neg b' \wedge a)),$$
$$\neg c' \wedge (\neg a' \vee d') \supset (\neg a' \vee (\neg b' \wedge a')).$$

On the other hand, we considered rules with variables and constants. We believe this extension is important as in applications, most programs have variables and many of them make use of constants. As we have seen above, this is a non-trivial extension even though the variables are eventually instantiated. It happens that for the problem of verifying strong equivalence, it remains to be co-NP-complete, but for normal equivalence, it becomes undecidable when rules have variables.

Historically, the translation that we gave above from logic programs to propositional theories has its root in a translation from logic programs to circumscriptive theories in [Lin, 1991]. Specifically, each rule of the form:

$$h \leftarrow p_1, \cdots, p_m, \text{not } p_{m+1}, \cdots, \text{not } p_n$$

is translated into the following sentence:

$$p_1 \wedge \cdots \wedge p_m \wedge \neg p'_{m+1} \wedge \cdots \wedge \neg p'_n \supset h.$$

For each program P, we will then get a propositional theory $T(P)$, and the original propositions in $T(P)$ are then circumscribed with the new "primed propositions" fixed. Finally, the equivalence $p \equiv p'$ for each proposition p is added to the result of the circumscription. It is shown [Lin, 1991] that each model of the final theory corresponds to a stable model of P, and vise versa.

As one can see, this translation is part of the one given in section 2 for programs without disjunctions and constraints. As it turns out, for capturing stable models, we could use the translation in section 2 as well:

$$Circum(\Gamma(P) \cup \Sigma; L; L') \wedge \bigwedge_{p \in L} (p \equiv p')$$

is equivalent to

$$Circum(T(P); L; L') \wedge \bigwedge_{p \in L} (p \equiv p'), \qquad (4)$$

where $L' = \{p' \mid p \in L\}$, $\Sigma = \{p \supset p' \mid p \in L\}$, and $Circum(W; A; B)$ denotes the circumscription of propositions in A in the theory W with propositions in B fixed.

[5] Turner [2001b] even allowed weight constraints.

This translation from normal logic programs to circumscription in [Lin, 1991] was originally derived from a translation from default logic to Lin and Shoham's logic of GK [Lin and Shoham, 1992], which is a preferential logic based on a bi-modal logic. A result by Lifschitz [1994] shows that disjunctive logic programs can be similarly captured in logic of GK, which implies that the aforementioned translation from logic programs to circumscriptive theories can be extended to disjunctive logic programs as well. In fact, it can be shown that if we let $T(P)$ to be the theory that contains formulas (2) and (3) for each rule (1) in P, then there will be a one-to-one correspondence between answer sets of P and models of (4).

6 Conclusions

We have investigated strong equivalence in logic programming using classical logic, and proved that the problem is co-NP-complete. There are several directions for future work. An important one is to collect a set of pairs of strongly equivalent programs that can be effectively used to optimize logic programs and queries.

7 Acknowledgements

I have benefited from discussing this work with Jicheng Zhao. I would also like to thank Vladimir Lifschitz and Hudson Turner for their useful comments on an earlier version of this paper. My special thanks to Hudson for giving me permission to include in this paper his translation from clauses to rules that does not use any constraints. Part of the work was done when the author was visiting the University of New South Wales and University of Western Sydney under the sponsorship of Norman Foo and Yan Zhang, respectively. This work was supported in part by the Research Grants Council of Hong Kong under Competitive Earmarked Research Grants HKUST6145/98E and HKUST6061/00E.

References

[Ebbinghaus and Flum, 1999] H. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, second edition, 1999.

[Gelfond and Lifschitz, 1991] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[Lifschitz *et al.*, 2001] V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, To appear, 2001.

[Lifschitz, 1994] V. Lifschitz. Minimal belief and negation as failure. *Artificial Intelligence*, 70:53–72, 1994.

[Lin and Shoham, 1992] F. Lin and Y. Shoham. A logic of knowledge and justified assumptions. *Artificial Intelligence*, 57:271–289, 1992.

[Lin, 1991] F. Lin. *A Study of Nonmonotonic Reasoning*. PhD thesis, Department of Computer Science, Stanford University, Stanford, CA, 1991.

[Maher, 1988] M. J. Maher. Equivalences of logic programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 627–658. Morgan Kaufmann Publishers, San Mateo, CA., 1988.

[Mendelson, 1987] E. Mendelson. *Introduction to Mathematical Logic*. Wadsworth & Brooks/Cole Advanced Books & Software, third edition, 1987.

[Niemelä, 1999] I. Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Ann. Math. and AI*, 25(3-4):241–273, 1999.

[Pearce *et al.*, 2001] D. Pearce, H. Tompits, and S. Woltran. Encodings for Equilibrium Logic and Logic Programs with Nested Expressions. In *Proc. EPIA-01*, pages 306–320, 2001.

[Sagiv, 1988] Y. Sagiv. Optimizing datalog programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 659–698. Morgan Kaufmann Publishers, San Mateo, CA., 1988.

[Shmueli, 1986] O. Shmueli. Decidability and expressiveness aspects of logic queries. In *Sixth ACM SIGACT-SIGMOD-SIGART Symposium on the Principles of database Systems*, pages 237–249, 1986.

[Turner, 2001a] H. Turner. Strong equivalence for logic programs and default theories (made easy). In *Proceedings of LPNMR'2001*, pages 81–92, 2001.

[Turner, 2001b] H. Turner. Strong equivalence made easy: nested expressions and weight constraints. Draft available on the author's web page, 2001.

Ultimate approximations in nonmonotonic knowledge representation systems

Marc Denecker
Department of Computer Science
K.U.Leeuven
Celestijnenlaan 200A, B-3001 Heverlee
Belgium

Victor W. Marek
Department of Computer Science
University of Kentucky
Lexington, KY, 40506-0046
USA

Mirosław Truszczyński
Department of Computer Science
University of Kentucky
Lexington, KY, 40506-0046
USA

Abstract

We study fixpoints of operators on lattices. To this end we introduce the notion of an approximation of an operator. We order approximations by means of a *precision* ordering. We show that each lattice operator O has a unique most precise or *ultimate* approximation. We demonstrate that fixpoints of this ultimate approximation provide useful insights into fixpoints of the operator O.

We apply our theory to logic programming and introduce the ultimate Kripke-Kleene, well-founded and stable semantics. We show that the ultimate Kripke-Kleene and well-founded semantics are more precise then their standard counterparts We argue that ultimate semantics for logic programming have attractive epistemological properties and that, while in general they are computationally more complex than the standard semantics, for many classes of theories, their complexity is no worse.

1 INTRODUCTION

Semantics of most knowledge representation languages are defined as collections of interpretations or possible-world structures. The sets of interpretations and possible-world structures, with some natural orderings, form complete lattices. Logic programs, and default and autoepistemic theories determine operators on these lattices. In many cases, semantics of programs and theories are given as fixpoints of these operators. Consequently, an abstract framework of lattices, operators on lattices and their fixpoints has emerged as a powerful tool in investigations of semantics of these logics. Studying semantics of nonmonotonic reasoning systems within an algebraic framework allows us to eliminate inessential details specific to a particular logic, simplify arguments and find common principles underlying different nonmonotonic formalisms.

The roots of this algebraic approach can be traced back to studies of semantics of logic programs [vEK76, AvE82, Fit85, Prz90] and of applications of lattices and bilattices in knowledge representation [Gin88]. Exploiting the concept of a bilattice and relying on some general properties of operators on lattices and bilattices, Fitting proposed an elegant algebraic treatment of all major 2-, 3- and 4-valued semantics of logic programs [Fit01], that is, the supported-model semantics [Cla78], stable-model semantics [GL88], Kripke-Kleene semantics [Fit85, Kun87] and well-founded semantics [VRS91].

In [DMT00a], we extended Fitting's work to a more abstract setting of the study of fixpoints of lattice operators. Central to our approach is the concept of an approximation of a lattice operator O. An approximation is an operator defined on a certain bilattice (the product of the lattice by itself, with two appropriately defined lattice orderings). Using purely algebraic techniques, for an approximation operator for O we introduced the notion of the stable operator and the concepts of the Kripke-Kleene, well-founded and stable fixpoints, and showed how they provide information about fixpoints of the operator O. In [DMT00a] we noted that our approach generalizes the results described in [Fit01]. We observed that the 4-valued immediate consequence operator $\mathcal{T}_P$ is an approximation operator for the 2-valued immediate consequence operator T_P and showed that all the semantics considered by Fitting can be derived from $\mathcal{T}_P$ by means of the general algebraic constructions that apply to *arbitrary* approximation operators.

In [DMT00b], we applied our algebraic approach to default and autoepistemic logics. Autoepistemic logic was defined by Moore [Moo84] to formalize the knowl-

edge of a rational agent with full introspection capabilities. In Moore's approach, an autoepistemic theory T defines a characteristic operator D_T on the lattice of all possible-world structures. Fixpoints of D_T (or, to be precise, their theories) are known as *expansions*. In [DMT00b], we proposed for D_T an approximation operator, $\mathcal{D}_T$, defined on a bilattice of belief pairs (pairs of possible-world structures). Complete fixpoints of $\mathcal{D}_T$ correspond to expansions of T (fixpoints of D_T), the least fixpoint of $\mathcal{D}_T$ provides a constructive approximation to all expansions (by analogy with logic programming, we called it the *Kripke-Kleene fixpoint*). Using general techniques introduced in [DMT00a] we derived from $\mathcal{D}_T$ its *stable* counterpart, the operator $\mathcal{D}_T^{st}$. Complete fixpoints of $\mathcal{D}_T^{st}$ yield a new semantics of *extensions* for autoepistemic logic. Finally, the least fixpoint of the stable operator results in yet another new semantics, the well-founded semantics for autoepistemic logic (again, called so due to analogies to the well-founded semantics in logic programming), which approximates all extensions.

The same picture emerged in the case of default logic [DMT00b]. For a default theory Δ we defined an operator $\mathcal{E}_\Delta$ and characterized all major semantics for default logic in terms of fixpoints of $\mathcal{E}_\Delta$. In particular, the standard semantics of extensions [Rei80] is determined by complete fixpoints of the stable operator $\mathcal{E}_\Delta^{st}$ derived from $\mathcal{E}_\Delta$. Our results on autoepistemic and default logics obtained in [DMT00b] allowed us to clarify the issue of their mutual relationship and provided insights into fundamental constructive principles underlying these two modes of nonmonotonic reasoning.

These result prove that the algebraic framework developed in [DMT00a] is an effective tool in studies of semantics of knowledge representation formalisms. It allowed us to establish a comprehensive semantic treatment for nonmonotonic logics and demonstrated that major nonmonotonic systems are closely related. However, the approach, as it was developed, is not entirely satisfactory. It provides no criteria that would allow us to prefer one approximation over another when attempting to define the concept of a stable fixpoint or when approximating fixpoints by means of the Kripke-Kleene or well-founded fixpoints. It does not give us any general indications how to obtain approximations and which approximation to pick. Thus, our theory leaves out a key link in the process of defining and approximating fixpoints of operators on lattices.

In particular, when defining semantics of nonmonotonic formalisms, we *select* an approximation operator, rather then derive it in a principled way. The approximations used, the bilattice operators $\mathcal{T}_P$, $\mathcal{D}_T$ and $\mathcal{E}_\Delta$, are not algebraically determined by their corresponding lattice operators T_P, D_T and E_Δ, respectively. Consequently, some programs or theories with the same basic operators have different Kripke-Kleene, well-founded or stable fixpoints associated with them.

We address this problem here. We extend our theory of approximations and introduce the notion of the precision of an approximation. We show that each lattice operator O has a unique *most precise* approximation which we call the *ultimate approximation* of O. Since the ultimate approximation is determined by O, it is well suited for investigations of fixpoints of O. As a result we obtain concepts of ultimate stable fixpoints, the ultimate Kripke-Kleene fixpoint and the ultimate well-founded fixpoint that depend on O only and not on a (possibly arbitrarily) selected approximation to O.

We apply our theory to logic programming, default logic and autoepistemic logic (only the first system is discussed here, due to space limitations). We compare ultimate semantics with the corresponding "standard" semantics of logic programs. In particular, we show that the ultimate Kripke-Kleene and the ultimate well-founded semantics are more precise then the standard Kripke-Kleene and well-founded semantics. This better accuracy comes, however, at a cost. We show that ultimate semantics are in general computationally more complex. On the other hand, we show that for wide classes of theories, including theories likely to occur in practice, the complexity remains the same. Thus, our new semantics may prove useful in computing stable models and default extensions.

The ultimate semantics have also properties that are attractive from the logic perspective. In particular, two programs or theories determining the same basic 2-valued operator have the same ultimate semantics. This property, as we noted, is not true in the standard case.

In summary, our contributions are as follows. We extend the algebraic theory of approximations by providing a principled way of deriving an approximation to a lattice operator. In this way, we obtain concepts of Kripke-Kleene fixpoint, well-founded fixpoint and stable fixpoints that are determined by the operator O and not by the choice of an approximation. In specific contexts of most commonly used nonmonotonic systems we obtain new semantics with desirable logical properties and possible computational applications.

2 PRELIMINARIES

Let $\langle L, \leq \rangle$ be a poset and let A be an operator on L. A poset is *chain-complete* if it contains the least element $\bot$ and if every chain of elements of L has a least upper bound (*lub*) in L. An element x of A is a *pre-fixpoint* of A if $A(x) \leq x$; x is a *fixpoint* of A if $A(x) = x$.

Let A be a monotone operator on a chain-complete poset $\langle L, \leq \rangle$. Let us define a sequence of elements of L by transfinite induction as follows: (1) $c^0 = \bot$; (2) $c^{\alpha+1} = A(c^\alpha)$; (3) $c^\alpha = lub(\{c^\beta : \beta < \alpha\})$, for a limit ordinal α. One can show that this sequence is well defined, that is has in L its least upper bound and that this least upper bound is the least fixpoint of A ($lfp(A)$, in symbols). One can also show that the least fixpoint of a monotone operator on a chain-complete poset is the least pre-fixpoint of A. That is, we have $lfp(A) = glb(\{x \in L : A(x) \leq x\})$. Monotone operators on chain-complete posets and their fixpoints and pre-fixpoints are discussed in [Mar76].

A lattice is a poset $\langle L, \leq \rangle$ such that $L \neq \emptyset$ and every pair of elements $x, y \in L$ has a unique greatest lower bound and least upper bound. A lattice is *complete* if its *every* subset has a greatest lower bound and a least upper bound. In particular, a complete lattice has a least and a greatest element denoted by $\bot$ and $\top$, respectively.

For any two elements $x, y \in L$, we define $[x, y] = \{z \in L : x \leq z \leq y\}$. If $\langle L, \leq \rangle$ is a complete lattice and $x \leq y$, then $\langle [x, y], \leq \rangle$ is a complete lattice, too.

Let $\langle L, \leq \rangle$ be a complete lattice. By the *product bilattice* [Gin88] of $\langle L, \leq \rangle$ we mean the set $L^2 = L \times L$ with the following two orderings $\leq_p$ and $\leq$:

1. $(x, y) \leq_p (x', y')$ if $x \leq x'$ and $y' \leq y$
2. $(x, y) \leq (x', y')$ if $x \leq x'$ and $y \leq y'$.

Both orderings are complete lattice orderings for L^2. However, in this paper we are mostly concerned with the ordering $\leq_p$.

An element $(x, y) \in L^2$ is *consistent* if $x \leq y$. We can think of a consistent element $(x, y) \in L^2$ as an *approximation* to every $z \in L$ such that $x \leq z \leq y$. With this interpretation in mind, the ordering $\leq_p$, when restricted to consistent elements, can be viewed as a *precision* ordering. Consistent pairs that are "higher" in the ordering $\leq_p$ provide tighter approximations. Maximal consistent elements with respect to $\leq_p$ are pairs of the form (x, x). We call approximations of the form (x, x) — *exact*.

We denote the set of all consistent pairs in L^2 by L^c. The set $\langle L^c, \leq_p \rangle$ is not a lattice. It is, however, chain-complete. Indeed, the element $(\bot, \top)$ is the least element in L^c and the following result shows that every chain in L^c has (in L^c) the least upper bound.

Proposition 2.1 *Let L be a complete lattice. If $\{(a^\alpha, b^\alpha)\}_\alpha$ is a chain of elements in $\langle L^c, \leq_p \rangle$ then $lub(\{a^\alpha\}_\alpha) \leq glb(\{a^\alpha\}_\alpha)$ and $(lub(\{a^\alpha\}_\alpha), glb(\{a^\alpha\}_\alpha)) = lub_{\leq_p}(\{(a^\alpha, b^\alpha)\}_\alpha)$.*

It follows that every $\leq_p$-monotone operator on L^c has a least fixpoint.

3 PARTIAL APPROXIMATIONS

For an operator $A : L^c \to L^c$, we denote by A^1 and A^2 its projections to the first and second coordinates, respectively. Thus, for every $(x, y) \in L^c$, we have $A(x, y) = (A^1(x, y), A^2(x, y))$. An operator $A : L^c \to L^c$ is a *partial approximation* operator if it is $\leq_p$-monotone and if for every $x \in L$, $A^1(x, x) = A^2(x, x)$. We denote the set of all partial approximation operators on L^c by $Appx(L^c)$. Let $A \in Appx(L^c)$. Since A is $\leq_p$-monotone and L^c is chain-complete, A has a least fixpoint, called the *Kripke-Kleene fixpoint* of A ($\mathrm{KK}(A)$, in symbols). Directly from the definition, it follows that $\mathrm{KK}(A)$ approximates all fixpoints of A.

If $A \in Appx(L^c)$ and $O : L \to L$ is an operator on L such that $A(x, x) = (O(x), O(x))$ then we say that A is a *partial approximation of O*. We denote the set of all partial approximations of O by $Appx(O)$. If A is a partial approximation of O then $x \in L$ is a fixpoint of O if and only if (x, x) is a fixpoint of A. Thus, for every fixpoint x of O, we have $\mathrm{KK}(A) \leq_p (x, x)$ or, equivalently, $\mathrm{KK}^1(A) \leq x \leq \mathrm{KK}^2(A)$, where $\mathrm{KK}^1(A)$ and $\mathrm{KK}^2(A)$ are the two components of the pair $\mathrm{KK}(A)$.

Operators from $Appx(L^c)$ describe ways to revise consistent approximations. Of particular interest are those situations when the revision of an approximation leads to another one that is at least as accurate. Let A be an operator on L^c. We call an approximation (a, b) *A-reliable* if $(a, b) \leq_p A(a, b)$.

Proposition 3.1 *Let L be a complete lattice and $A \in Appx(L^c)$. If $(a, b) \in L^c$ is A-reliable then, for every $x \in [\bot, b]$, $A^1(x, b) \in [\bot, b]$ and, for every $x \in [a, \top]$, $A^2(a, x) \in [a, \top]$.*

Proof: Let $x \in [\bot, b]$. Then $(x, b) \leq_p (b, b)$. By the $\leq_p$-monotonicity of A,

$$A^1(x, b) \leq A^1(b, b) = A^2(b, b) \leq A^2(a, b) \leq b.$$

The last inequality follows from the fact that (a, b) is A-reliable. The second part of the assertion can be proved in a similar manner. □

This proposition implies that for every A-reliable pair (a,b), the restrictions of $A^1(\cdot,b)$ to $[\bot,b]$ and $A^2(a,\cdot)$ to $[a,\top]$ are in fact operators on $[\bot,b]$ and $[a,\top]$, respectively. Moreover, they are $\leq$-monotone operators on the posets $\langle[\bot,b],\leq\rangle$ and $\langle[a,\top],\leq\rangle$. Since $\langle[\bot,b],\leq\rangle$ and $\langle[a,\top],\leq\rangle$ are complete lattices, the operators $A^1(\cdot,b)$ and $A^2(a,\cdot)$ have least fixpoints in the lattices $\langle[\bot,b],\leq\rangle$ and $\langle[a,\top],\leq\rangle$, respectively. We define:

$$b^{A\downarrow} = lfp(A^1(\cdot,b)) \quad \text{and} \quad a^{A\uparrow} = lfp(A^2(a,\cdot)).$$

We call the mapping $(a,b) \mapsto (b^{A\downarrow}, a^{A\uparrow})$, defined on the set of A-reliable elements of L^c, the *stable revision operator for* A. When A is clear from the context, we will drop the reference to A from the notation.

Directly from the definition of the stable revision operator it follows that for every A-reliable pair, $b^\downarrow \leq b$ and $a \leq a^\uparrow$.

The stable revision operator for $A \in Appx(L^c)$ is crucial. It allows us to distinguish an important subclass of the class of all fixpoints of A. Let L be a complete lattice and let $A \in Appx(L^c)$. We say that $(x,y) \in L^c$ is a *stable fixpoint* of A if (x,y) is A-reliable and is a fixpoint of the stable revision operator (that is, $x = y^\downarrow$ and $y = x^\uparrow$). By the A-reliability of (x,y), the second requirement is well defined.

Stable fixpoints of an operator are, in particular, its fixpoints.

Proposition 3.2 *Let L be a complete lattice and let $A \in Appx(L^c)$. If (x,y) is a stable fixpoint of A then (x,y) is a fixpoint of A.*

Proof: Since (x,y) is stable, $x = lfp(A^1(\cdot,y))$. In particular, $x = A^1(x,y)$. Similarly, $y = A^2(x,y)$. □

Let O be an operator on a complete lattice L and let $A \in Appx(O)$. We say that x is an *A-stable* fixpoint of O if (x,x) is a stable fixpoint of A. The notation is justified. Indeed, it follows from Proposition 3.2 and our earlier remarks that every stable fixpoint of O is, in particular, a fixpoint of O.

The notion of A-reliability is not strong enough to guarantee desirable properties of the stable revision operator. In particular, if $(a,b) \in L^c$ is A-reliable, it is not true in general that $(b^\downarrow, a^\uparrow)$ is consistent nor that $(a,b) \leq_p (b^\downarrow, a^\uparrow)$. There is, however, a class of A-reliable pairs for which both properties hold. An A-reliable approximation (a,b) is *A-prudent* if $a \leq b^\downarrow$. We note that every stable fixpoint of A is A-prudent. We will now prove several basic properties of A-prudent approximations.

Proposition 3.3 *Let L be a complete lattice, $A \in Appx(L^c)$ and $(a,b) \in L^c$ be A-prudent. Then, $(b^\downarrow, a^\uparrow)$ is consistent, A-reliable and A-prudent and $(a,b) \leq_p (b^\downarrow, a^\uparrow)$.*

Proof: By the definition of $b^\downarrow$ and $a^\uparrow$ we have that $b^\downarrow \leq b$ and $a \leq a^\uparrow$. Moreover, since (a,b) is A-prudent, it follows that $a \leq b^\downarrow$.

Next, since (a,b) is A-reliable, it follows that $a \leq b$ and $A^2(a,b) \leq b$. Thus, b is a pre-fixpoint of $A^2(a,\cdot)$. Consequently, $a^\uparrow \leq b$ (as $a^\uparrow$ is the least fixpoint of $A^2(a,\cdot)$). Hence, $(a,b) \leq_p (b^\downarrow, a^\uparrow)$.

By the $\leq_p$-monotonicity of A we obtain:

$$A^1(a^\uparrow, b) \leq A^1(a^\uparrow, a^\uparrow) = A^2(a^\uparrow, a^\uparrow) \leq A^2(a, a^\uparrow) = a^\uparrow.$$

It follows that $a^\uparrow$ is a pre-fixpoint of the operator $A^1(\cdot,b)$. Thus, $b^\downarrow = lfp(A^1(\cdot,b)) \leq a^\uparrow$ and so, $(b^\downarrow, a^\uparrow)$ is consistent.

Let us now observe that $b^\downarrow = A^1(b^\downarrow, b) \leq A^1(b^\downarrow, a^\uparrow)$. Similarly, $a^\uparrow = A^2(a, a^\uparrow) \geq A^2(b^\downarrow, a^\uparrow)$. Thus, the pair $(b^\downarrow, a^\uparrow)$ is reliable.

Lastly, we note that for every $x \in [\bot, a^\uparrow]$, $A^1(x,b) \leq A^1(x, a^\uparrow) \leq a^\uparrow$ (the last inequality follows by the A-reliability of $(b^\downarrow, a^\uparrow)$). Hence, $b^\downarrow = lfp(A^1(\cdot,b)) \leq lfp(A^1(\cdot, a^\uparrow))$ and, consequently, $(b^\downarrow, a^\uparrow)$ is A-prudent. □

Let us observe that an A-reliable pair (a,b) is revised by an operator A into a more accurate approximation $A(a,b)$. An A-prudent pair (a,b) can be revised "even more". Namely, it is easy to see that $A^1(a,b) \leq A^1(b^\downarrow, b) = b^\downarrow$ and $a^\uparrow = A^2(a, a^\uparrow) \leq A^2(a,b)$. Thus, $A(a,b) \leq_p (b^\downarrow, a^\uparrow)$. In other words, $(b^\downarrow, a^\uparrow)$ is indeed at least as precise revision of (a,b) as $A(a,b)$ is.

The stable revision operator satisfies a certain monotonicity property.

Proposition 3.4 *Let L be a complete lattice and let $A \in Appx(L^c)$. If $(a,b) \in L^c$ is A-reliable, $(c,d) \in L^c$ is A-prudent and if $(a,b) \leq_p (c,d)$, then $(b^\downarrow, a^\uparrow) \leq_p (d^\downarrow, c^\uparrow)$.*

Proof: Clearly, we have $d^\downarrow \leq c^\uparrow \leq d \leq b$. By the $\leq_p$-monotonicity of A, it follows that $A^1(d^\downarrow, b) \leq A^1(d^\downarrow, d) = d^\downarrow$. Thus, $d^\downarrow$ is a pre-fixpoint of $A^1(\cdot,b)$. Since $b^\downarrow$ is the least fixpoint of $lfp(A^1(\cdot,b))$, it follows that $b^\downarrow \leq d^\downarrow$.

It remains to prove that $c^\uparrow \leq a^\uparrow$. Let $u = glb(a^\uparrow, d^\downarrow)$. By Proposition 3.3, $(c,d) \leq_p (d^\downarrow, c^\uparrow)$. Since $(a,b) \leq_p (c,d)$, it follows that $a \leq d^\downarrow$. Further, by the A-reliability of (a,b) and (c,d), we have $a \leq a^\uparrow$ and

$d^{\downarrow} \leq d$. Thus, $a \leq u \leq a^{\uparrow}$ and $u \leq d^{\downarrow} \leq d$. Consequently,

$$A^1(u,d) \leq A^1(u,u) = A^2(u,u) \leq A^2(a,a^{\uparrow}) = a^{\uparrow}$$

and

$$A^1(u,d) \leq A^1(d^{\downarrow},d) = d^{\downarrow}.$$

It follows that $A^1(u,d) \leq glb(a^{\uparrow},d^{\downarrow}) = u$. In particular, u is a pre-fixpoint of $A^1(\cdot,d)$. Since $d^{\downarrow}$ is the least fixpoint of $A^1(\cdot,d)$, $d^{\downarrow} \leq u$. Hence, $d^{\downarrow} \leq a^{\uparrow}$.

We now have $a \leq c \leq d^{\downarrow} \leq a^{\uparrow}$ (the first inequality follows from the assumption $(a,b) \leq (c,d)$, the second one follows by Proposition 3.3 from the assumption that (c,d) is A-prudent). Thus, $a \leq c \leq a^{\uparrow}$ and the $\leq_p$-monotonicity of A implies

$$A^2(c,a^{\uparrow}) \leq A^2(a,a^{\uparrow}) = a^{\uparrow}.$$

Hence, $a^{\uparrow}$ is a pre-fixpoint of $A^2(c,\cdot)$. Since $c^{\uparrow}$ is the least fixpoint of $A^2(c,\cdot)$, it follows that $c^{\uparrow} \leq a^{\uparrow}$. □

Since stable fixpoints are prudent, we obtain the following corollary.

Corollary 3.5 *Let L be a complete lattice, $A \in Appx(L^c)$ and let $(c,d) \in L^c$ be a stable fixpoint of A. If $(a,b) \in L^c$ is A-reliable and $(a,b) \leq_p (c,d)$ then $(b^{\downarrow},a^{\uparrow}) \leq_p (c,d)$.* □

The next result states that the limit of a chain of A-prudent pairs is A-prudent.

Proposition 3.6 *Let L be a complete lattice, $A \in Appx(L^c)$ and let $\{(a^{\alpha},b^{\alpha})\}_{\alpha}$ be a chain of A-prudent pairs from L^c. Then, $lub(\{(a^{\alpha},b^{\alpha})\}_{\alpha})$ is A-prudent.*

Proof: Let us set $a^{\infty} = lub(\{a^{\alpha}\}_{\alpha})$ and $b^{\infty} = glb(\{b^{\alpha}\}_{\alpha})$. By Proposition 2.1, (a^{∞},b^{∞}) is consistent and $(a^{\infty},b^{\infty}) = lub(\{(a^{\alpha},b^{\alpha})\}_{\alpha})$. Let us now observe that, by A-reliability of (a^{α},b^{α}) and $\leq_p$-monotonicity of A, we have $(a^{\alpha},b^{\alpha}) \leq_p A(a^{\alpha},b^{\alpha}) \leq_p A(a^{\infty},b^{\infty})$. Thus, $(a^{\infty},b^{\infty}) = lub(\{(a^{\alpha},b^{\alpha})\}_{\alpha}) \leq A(a^{\infty},b^{\infty})$. It follows that (a^{∞},b^{∞}) is A-reliable.

The A-reliability of (a^{∞},b^{∞}) implies, in particular, that for every $x \in [\bot,b^{\infty}]$, $A^1(x,b^{\infty}) \leq b^{\infty}$. Thus, by $\leq_p$-monotonicity of A, for every $x \in [\bot,b^{\infty}]$

$$A^1(x,b^{\alpha}) \leq A^1(x,b^{\infty}) \leq b^{\infty}.$$

Hence, pre-fixpoints of $A^1(\cdot,b^{\infty})$ are prefixpoints of $A^1(\cdot,b^{\alpha})$ and, consequently,

$$lfp(A^1(\cdot,b^{\alpha})) \leq lfp(A^1(\cdot,b^{\infty})).$$

Since (a^{α},b^{α}) is A-prudent, we have that $a^{\alpha} \leq lfp(A^1(\cdot,b^{\alpha}))$. Thus, for arbitrary α, $a^{\alpha} \leq lfp(A^1(\cdot,b^{\infty}))$ and, consequently, $a^{\infty} \leq lfp(A^1(\cdot,b^{\infty}))$. It follows that (a^{∞},b^{∞}) is A-prudent. □

We will now prove that the set of all stable fixpoints of an operator has a least element (in particular, it is not empty). To this end, we define a sequence $\{(a^{\alpha},b^{\alpha})\}_{\alpha}$ of elements of L^c by transfinite induction:

1. $(a^0,b^0) = (\bot,\top)$
2. If $\alpha = \beta + 1$, we define $a^{\alpha} = b^{\beta\downarrow}$ and $b^{\alpha} = a^{\beta\uparrow}$
3. If α is a limit ordinal, we define $(a^{\alpha},b^{\alpha}) = lub(\{(a^{\beta},b^{\beta}) : \beta < \alpha\})$.

Theorem 3.7 *The sequence $\{(a^{\alpha},b^{\alpha})\}_{\alpha}$ is well defined, $\leq_p$-monotone and its limit is the least stable fixpoint of a partial approximation operator A.*

Proof: It is obvious that $(\bot,\top)$ is A-prudent. Thus, by the transfinite induction it follows that each element in the sequence is well defined and A-prudent (Propositions 3.3 and 3.6 settle the cases of successor ordinals and limit ordinals, respectively). In the same way, one can establish the $\leq_p$-monotonicity of the sequence.

Let $(a^{\infty},b^{\infty}) = lub(\{(a^{\beta},b^{\beta})\}_{\alpha})$. By Proposition 3.6, (a^{∞},b^{∞}) is A-prudent. Thus, (a^{∞},b^{∞}) is A-reliable. Moreover, we have $a^{\infty} = (b^{\infty})^{\downarrow}$ and $b^{\infty} = (a^{\infty})^{\uparrow}$. Thus, (a^{∞},b^{∞}) is a stable fixpoint of A. Further, it is easy to see by transfinite induction and Corollary 3.5 that (a^{∞},b^{∞}) approximates all stable fixpoints of A. Thus, it is the least stable fixpoint of A. □

We call this least stable fixpoint the *well-founded fixpoint* of A and denote it by $\mathrm{WF}(A)$. The well-founded fixpoint approximates all stable fixpoints of A. In particular, it approximates all A-stable fixpoints of the operator O. That is, for every A-stable fixpoint x of O, $\mathrm{WF}(A) \leq_p (x,x)$ or, equivalently, $\mathrm{WF}^1(A) \leq x \leq \mathrm{WF}^2(A)$, where $\mathrm{WF}^1(A)$ and $\mathrm{WF}^2(A)$ are the two components of the pair $\mathrm{WF}(A)$. Moreover, the well-founded fixpoint is more precise than the Kripke-Kleene fixpoint: for $A \in Appx(O)$, $\mathrm{KK}(A) \leq_p \mathrm{WF}(A)$.

In [DMT00b, DMT00a], we showed that when applied to appropriately chosen approximation operators in logic programming, default logic and autoepistemic logic, these algebraic concepts of fixpoints, stable fixpoints, the Kripke-Kleene fixpoint and the well-founded fixpoint provide all major semantics for these nonmonotonic systems and allow us to understand their interrelations.

We need to emphasize that the concept of a partial approximation introduced here is different from the concept of *approximation* introduced in [DMT00a]. The

latter notion is defined as an operator of the whole bilattice L^2. That choice was motivated by our search for generality and potential applications of inconsistent fixpoints in situations when we admit a possibility of some statements being overdefined. While different, both approaches are very closely related[1].

4 ULTIMATE APPROXIMATIONS

Partial approximations in $Appx(L^c)$ can be ordered. Let $A, B \in Appx(L^c)$. We say that A is *less precise* than B ($A \leq_p B$, in symbols) if for each pair $(x, y) \in L^c$, $A(x, y) \leq_p B(x, y)$. It is easy to see that if $A \leq_p B$ then there is an operator O on the lattice L such that $A, B \in Appx(O)$.

Lemma 4.1 *Let L be a complete lattice and $A, B \in Appx(L^c)$. If $A \leq_p B$ and $(a, b) \in L^c$ is A-prudent then (a, b) is B-prudent and $(b^{A\downarrow}, a^{A\uparrow}) \leq_p (b^{B\downarrow}, a^{B\uparrow})$.*

Proof: Clearly, $(a, b) \leq_p A(a, b) \leq B(a, b)$. Thus, (a, b) is B-reliable.

For each pre-fixpoint $x \leq b$ of $B^1(\cdot, b)$, $A^1(x, b) \leq B^1(x, b) \leq x$. Consequently, x is a prefixpoint of $A^1(\cdot, b)$. It follows that $b^{A\downarrow} \leq b^{B\downarrow}$. Since $a \leq b^{A\downarrow}$, $a \leq b^{B\downarrow}$. Thus (a, b) is B-prudent.

Likewise, we can prove that any pre-fixpoint of $A^2(a, \cdot)$ is a prefixpoint of $B^2(a, \cdot)$, and consequently, $a^{B\uparrow} \leq a^{A\uparrow}$. Since also $b^{A\downarrow} \leq b^{B\downarrow}$, it follows that $(b^{A\downarrow}, a^{A\uparrow}) \leq_p (b^{B\downarrow}, a^{B\uparrow})$. □

More precise approximation have more precise Kripke-Kleene and well-founded fixpoints.

Theorem 4.2 *Let O be an operator on a complete lattice L. Let $A, B \in Appx(O)$. If $A \leq_p B$ then* $\mathrm{KK}(A) \leq_p \mathrm{KK}(B)$ *and* $\mathrm{WF}(A) \leq_p \mathrm{WF}(B)$.

Proof: Let us denote by $\{(a^\alpha_A, b^\alpha_A)\}_\alpha$ the sequence of elements of $\langle L^c, \leq_p\rangle$ obtained by iterating the operator A over $(\bot, \top)$. The sequence $\{(a^\alpha_B, b^\alpha_B)\}_\alpha$ is defined in the same way. Since $A \leq_p B$, it follows by an easy induction that for every ordinal α, $(a^\alpha_A, b^\alpha_A) \leq_p (a^\alpha_B, b^\alpha_B)$. Since $\mathrm{KK}(A)$ is the limit of the sequence $\{(a^\alpha_A, b^\alpha_A)\}_\alpha$ and $\mathrm{KK}(B)$ is the limit of the sequence $\{(a^\alpha_B, b^\alpha_B)\}_\alpha$, it follows that $\mathrm{KK}(A) \leq_p \mathrm{KK}(B)$.

To prove the second part of the assertion, we will now assume that the sequences $\{(a^\alpha_A, b^\alpha_A)\}_\alpha$ and $\{(a^\alpha_B, b^\alpha_B)\}_\alpha$ denote the sequences used in the definition of the well-founded fixpoints of A and B, respectively. To prove the assertion we will now show that for every ordinal α, $(a^\alpha_A, b^\alpha_A) \leq_p (a^\alpha_B, b^\alpha_B)$.

Clearly, $(a^0_A, b^0_A) \leq_p (a^0_B, b^0_B)$. Let us assume that $\alpha = \beta + 1$ and that $(a^\beta_A, b^\beta_A) \leq_p (a^\beta_B, b^\beta_B)$. Since (a^β_A, b^β_A) is A-prudent, Lemma 4.1 entails that it is B-prudent and

$$(a^\alpha_A, b^\alpha_A) = ((b^\beta_A)^{A\downarrow}, (a^\beta_A)^{A\uparrow}) \leq_p ((b^\beta_A)^{B\downarrow}, (a^\beta_A)^{B\uparrow}).$$

By Proposition 3.4,

$$((b^\beta_A)^{B\downarrow}, (a^\beta_A)^{B\uparrow}) \leq_p ((b^\beta_B)^{B\downarrow}, (a^\beta_B)^{B\uparrow}) = (a^\alpha_B, b^\alpha_B).$$

The case of the limit ordinal α is straightforward.

Since $\mathrm{WF}(A)$ and $\mathrm{WF}(B)$ are the limits of the sequences $\{(a^\alpha_A, b^\alpha_A)\}_\alpha$ and $\{(a^\alpha_B, b^\alpha_B)\}_\alpha$, respectively, the second part of the assertion follows. □

The next result shows that as the precision of an approximation grows, all exact fixpoints and exact stable fixpoints are preserved.

Theorem 4.3 *Let O be an operator on a complete lattice L. Let $A, B \in Appx(O)$. If $A \leq_p B$ then every exact fixpoint of A is an exact fixpoint of B, and every exact stable fixpoint of A (that is, an A-stable fixpoint of O) is also an exact stable fixpoint of B (that is, a B-stable fixpoint of O).*

Proof: Since for every $x \in L$, $A(x, x) = B(x, x) = (O(x), O(x))$, the first part of the assertion follows. Let us now assume that (x, x) is an exact stable fixpoint of A. In particular, it follows that (x, x) is a fixpoint of A and is A-prudent. By Lemma 4.1, (x, x) is B-prudent and $(x, x) \leq_p (x^{B\downarrow}, x^{B\uparrow})$. The latter pair is consistent (Proposition 3.3). Consequently, (x, x) is $(x^{B\downarrow}, x^{B\uparrow})$ and hence x is an exact stable fixpoint of B. □

Non-exact fixpoints are not preserved, in general. Let us consider two partial approximations A and B such that $A \leq_p B$. Let us also assume that $\mathrm{WF}(A) <_p \mathrm{WF}(B)$ (that is, A has a strictly less precise well-founded fixpoint than B). Then, clearly, $\mathrm{WF}(A)$ is no longer a stable fixpoint of B. Thus, fixpoints of A may disappear when we move on to a more precise approximation B.

More precise approximations of a non-monotone operator O yield more precise well-founded fixpoints and additional exact stable fixpoints. The natural question is whether there exists an *ultimate approximation* of O, that is, a partial approximation most precise with respect to the ordering $\leq_p$. Such approximation would have a most precise Kripke-Kleene and well-founded

[1] We will include a detailed discussion of the relationship between the two approaches in the full version of the paper.

fixpoint and a largest set of exact stable fixpoints. We will show that the answer to this key question is positive. Such ultimate approximation, being a distinguished object in the collection of all approximations can be viewed as determined by O. Consequently, fixpoints of the ultimate approximation of O (including stable, Kripke-Kleene and well-founded fixpoints) can be regarded as determined by O and can be associated with it.

We start by providing a non-constructive argument for the existence of ultimate approximations. Let us note that the set $Appx(O)$ is not empty. Indeed, let us define $A_O(x,y) = (O(x), O(x))$, if $x = y$, and $A_O(x,y) = (\bot, \top)$, otherwise. It is easy to see that $A_O \in Appx(O)$ and that it is the least precise element in $Appx(O)$. Next, we observe that $Appx(O)$ with the ordering $\leq_p$ is a complete lattice, as the set $Appx(O)$ is closed under the operations of taking greatest lower bounds and least upper bounds. It follows that $Appx(O)$ has a greatest element (most precise approximation). We call this partial approximation the *ultimate approximation* of O and denote it by U_O.

We call the Kripke-Kleene and the well-founded fixpoints of U_O, the *ultimate Kripke-Kleene* and the *ultimate well-founded fixpoint* of O. We denote them by $\mathrm{KK}(O)$ and $\mathrm{WF}(O)$, respectively. We call a stable fixpoint of U_O an *ultimate partial stable fixpoint* of O. We refer to an *exact* stable fixpoint of U_O as an *ultimate stable fixpoint* of O. Exact fixpoints of all partial approximations are the same and correspond to fixpoints of O. Thus, there is no need to introduce the concept of an ultimate exact fixpoint of O. We have the following corollary to Theorems 4.2 and 4.3.

Corollary 4.4 *Let O be an operator on a complete lattice L. For every $A \in Appx(O)$, $\mathrm{KK}(A) \leq_p \mathrm{KK}(U_O)$, $\mathrm{WF}(A) \leq_p \mathrm{WF}(U_O)$ and every A-stable fixpoint of O is an ultimate stable fixpoint of O.*

We will now provide a constructive characterization of the notion. To state the result, for every $x, y \in L$ such that $x \leq y$, we define $O([x,y]) = \{O(z) : z \in [x,y]\}$.

Theorem 4.5 *Let O be an operator on a complete lattice L. Then, for every $(x,y) \in L^c$, $U_O(x,y) = (glb(O([x,y])), lub(O([x,y])))$.*

Proof: We define an operator $C : L^c \to L^2$ by setting

$$C(x,y) = (glb(O([x,y])), lub(O([x,y]))).$$

First, let us notice that since $glb(O([x,y])) \leq lub(O([x,y]))$, the operator C maps L^c into L^c. Moreover, it is easy to see that C is $\leq_p$-monotone. Lastly, since $O([x,x]) = \{O(x)\}$,

$$glb(O([x,x])) = lub(O([x,x])) = O(x).$$

and, consequently, $C(x,x) = (O(x), O(x))$. Thus, it follows that C is a partial approximation of O. Since U_O is the most precise approximation, we have $C \leq_p U_O$.

On the other hand, $U_O(x,y) \leq (O(z), O(z))$ for every $z \in [x,y]$. Therefore $U_O^1(x,y) \leq_p O(z)$ for all $z \in [x,y]$ and thus $U_O^1(x,y) \leq glb(O([x,y]))$. Similarly, $lub(O([x,y])) \leq U_O^2(x,y)$. Since $x \leq y$ are arbitrary, $U_O \leq_p C$, as desired. □

With this result we obtain an explicit characterization of ultimate stable fixpoints of an operator O.

Corollary 4.6 *Let L be a complete lattice. An element $x \in L$ is an ultimate stable fixpoint of an operator $O : L \to L$ if and only if x is the least fixpoint of the operator $glb(O([\cdot, x]))$ regarded as an operator on $[\bot, x]$.*

We conclude this section by describing ultimate approximations for monotone and antimonotone operators on L.

Proposition 4.7 *If O is a monotone operator on a complete lattice L then for every $(x,y) \in L^c$, $U_O(x,y) = (O(x), O(y))$. If O is antimonotone then for every $(x,y) \in L^c$, $U_O(x,y) = (O(y), O(x))$.*

Proof: By Theorem 4.5,

$$U_O(x,y) = (glb(O([x,y])), lub(O([x,y]))).$$

Now, it is easy to see that if O is monotone, then $glb(O([x,y])) = O(x)$ and $lub(O([x,y])) = O(y)$. If O is antimonotone, then $glb(O([x,y])) = O(y)$ and $lub(O([x,y])) = O(x)$. The proposition follows. □

Using the results from [DMT00a] and Proposition 4.7 we now obtain the following corollary.

Corollary 4.8 *Let O be an operator on a complete lattice L. If O is monotone, then the least fixpoint of O is the ultimate well-founded fixpoint of O and the unique ultimate stable fixpoint of O. If O is antimonotone, then $\mathrm{KK}(O) = \mathrm{WF}(O)$ and every fixpoint of O is an ultimate stable fixpoint of O.*

5 ULTIMATE SEMANTICS FOR LOGIC PROGRAMMING

The basic operator in logic programming is the one-step provability operator T_P introduced in [vEK76]. It

is defined on the lattice of all interpretations. This lattice consists of subsets of the set of all atoms appearing in P and is ordered by inclusion (we identify truth assignments with subsets of atoms that are assigned the value t).

Let P be a logic program. We denote by U_P the ultimate approximation operator for the operator T_P. By specializing Theorem 4.5 to the operator T_P we obtain that for every two interpretations $I \subseteq J$,

$$U_P(I,J) = (glb(T_P([I,J])), lub(T_P([I,J]))).$$

Replacing the ultimate approximation operator U_O in the definitions of ultimate Kripke-Kleene, well-founded and stable fixpoints with U_P results in the corresponding notions of ultimate Kripke-Kleene, well-founded and stable models (semantics) of a program P.

We are now in a position to discuss commonsense reasoning intuitions underlying abstract algebraic concepts of ultimate approximation and its fixpoints. Let us consider two interpretations I and J such that $I \subseteq J$. We interpret I as a current lower bound and J as a current upper bound on the set of atoms that are true (under P). Thus, I specifies atoms that are definitely true, while J specifies atoms that are possibly true. Arguably, if an atom p is derived by applying the operator T_P to *every* interpretation $K \in [I,J]$, it can safely be assumed to be true (in the context of the knowledge represented by I and J). Thus, the set $I' = glb(T_P([I,J]))$ can be viewed as a revision of I.

Similarly, since every interpretation $K \in [I,J]$ must be regarded as possible according to the pair (I,J) of conservative and liberal estimates, an atom might possibly be true if it can be derived by the operator T_P from *at least* one interpretation in $[I,J]$. Thus, the set $J' = lub(T_P([I,J]))$, consisting of all such atoms, can be regarded as a revision of J. Clearly, $(I',J') = U_P(I,J)$ and, consequently, U_P can be viewed as a way to revise our knowledge about the logical values of atoms as determined by a program P from (I,J) to (I',J').

By iterating U_P starting at $(\bot,\top)$, we obtain the ultimate Kripke-Kleene model of P as an approximation that cannot be further improved by applying U_P. The ultimate Kripke Kleene model of P approximates all fixpoints of U_P and, in particular, all supported models of P. Often, however, the Kripke-Kleene model is too weak as we are commonly interested in those (partial) models of P that satisfy some minimality or groundedness conditions. These requirements are satisfied by ultimate stable models and, in particular, by the ultimate well-founded model of P.

When constructing the ultimate well-founded model, we start by assuming no knowledge about the status of atoms: no atom is known true and all atoms are assumed possible. Our goal is to improve on these bounds.

To improve on the lower bound, we proceed as follows. Our current knowledge does not preclude any interpretation and all of them (the whole segment $[\bot,\top]$) need to be taken into account. If some atom p can be derived by applying the operator T_P to each element of $[\bot,\top]$ then, arguably, p could be accepted as definitely true. The set of all these atoms is exactly $glb(T_P([\bot,\top]))$. So, this set, say I_1, can be taken as a safe new lower bound, giving a smaller interval $[I_1,\top]$ of possible interpretations. We now repeat the same process and obtain a new lower bound, say I_2, consisting of those atoms that can be derived from every interpretation in $[I_1,\top]$. It is given by $I_2 = glb(T_P([I_1,\top]))$. Clearly, I_2 improves on I_1. We iterate this process until a fixpoint is reached. This fixpoint, say I^1, consists of all these atoms for which there is a *constructive* argument that they are true, given that no atoms are known to be false (all atoms are possible). Thus, it provides a safe lower bound for the set of atoms the program should specify as true.

The reasoning for revising the upper bound is different. The goal is to make false all atoms for which there cannot be a constructive argument that they are true. Let us consider an interpretation J such that for every $K \in [\bot,J]$, $T_P(K) \in [\bot,J]$, or equivalently, $lub(T_P([\bot,J])) \subseteq J$. An atom $p \notin J$ (false in J) cannot be made true by applying T_P to any element in the segment $[\bot,J]$. In order to derive p by means of T_P, some atoms that are false in J would have to be made true. That, however, would mean that p is not grounded and could be assumed to be false. Thus, each such interpretation J represents an upper estimate on what is possible (its complement gives a lower estimate on what is false) under the assumption that no atom is known to be true yet. It turns out that there is a least interpretation, say J^1 such that $lub(T_P([\bot,J^1])) \subseteq J^1$ and it can be constructed in a bottom up way by iterating the operator $lub(T_P([\bot,\cdot]))$. This interpretation can be taken as a safe lower bound on what is false (given that no atom is known to be true).

The pair (I^1,J^1) is the first improvement on $(\bot,\top)$. It is precisely the pair produced by the first iteration of the general well-founded fixpoint definition given earlier. It can now be used, in place of $(\bot,\top)$, to obtain an even more refined estimate, (I^2,J^2) and the process continues until the fixpoint is reached. The resulting pair is the ultimate well-founded model of P.

This discussion demonstrates that abstract algebraic concepts of ultimate approximations can be given a sound intuitive account.

We will now discuss the properties of the ultimate semantics for logic programs.

Theorem 5.1 *Let P, P' be two programs such that $T_P = T_{P'}$. Then, the ultimate well-founded models and ultimate stable models of P and P' coincide.*

Proof: Theorem 4.5 implies that $U_P = U_{P'}$. But then all fixpoints of U_P and $U_{P'}$ coincide. Thus, the result follows. □

This assertion does not hold for the (standard) well-founded and stable models. For instance, let $P_1 = \{p \leftarrow p,\ \ p \leftarrow \neg p\}$ and $P_2 = \{p \leftarrow\}$. Clearly, $T_{P_1} = T_{P_2}$. However, P_2 has a stable model, $\{p\}$, while P_1 has no stable models. Furthermore, p is true in the well-founded model of P_2 and unknown in the well-founded model of P_1.

Another appealing property is that the ultimate well-founded model of a program P with monotone operator T_P is the least fixpoint of this operator (the least model of P). This is a corollary of Proposition 4.8. It is not satisfied by the standard well-founded semantics, as shown by the program P_1.

In many cases, the ultimate well-founded semantics coincides with the standard well-founded semantics. A consequence of Corollary 4.4 is that if the well-founded model of a program is two-valued, then it coincides with the ultimate well-founded model. Thus, we have the following result dealing with the classes of Horn and weakly stratified programs [Prz90]:

Proposition 5.2 *If a logic program P is a Horn program or a (weakly) stratified program, then its ultimate well-founded semantics coincides with the standard well-founded semantics.*

Proof: Let P be a Horn program or a weakly stratified program (the argument is the same). Let WF_P be the well-founded model of P. Let T_P be the van Emden-Kowalski operator for P, and let $\mathcal{T}_P$ be the corresponding 3-valued operator [Fit85]. Then, $\mathcal{T}_P$ is an approximation of T_P and the well-founded model of P satisfies $\mathrm{WF}_P = \mathrm{WF}(\mathcal{T}_P)$ [DMT00a]. Moreover, for weakly stratified programs, WF_P is two-valued [VRS91]. By Corollary 4.4

$$\mathrm{WF}_P = \mathrm{WF}(\mathcal{T}_P) \leq_p \mathrm{WF}(U_P).$$

Since $\mathrm{WF}(U_P)$ is consistent, and WF_P is complete, it follows that $\mathrm{WF}_P = \mathrm{WF}(U_P)$, as required. □

We now show that in general, attractive properties of ultimate semantics come at a price. Namely, we have the following two theorems.

Theorem 5.3 *The problem "given a finite propositional logic program P, decide whether P has a complete ultimate stable model" is Σ_2^P-complete.*

Theorem 5.4 *The problems "given a finite propositional logic program, compute the ultimate well-founded fixpoint of P" and "given a finite propositional logic program, compute the ultimate Kripke-Kleene fixpoint of P" are in the class Δ_2^P.*

These results might put in doubt the usefulness of ultimate semantics. However, for wide classes of programs the complexity does not grow. Let k be a fixed integer. We define the class $\mathcal{E}_k$ to consist of all logic programs P such that for every atom $p \in At(P)$ at least one of the following conditions holds:

1. P contains at most k clauses with p as the head;
2. the body of each clause with the head p consists of at most two elements;
3. the body of each clause with the head p contains at most one positive literal;
4. the body of each clause with the head p contains at most one negative literal.

Theorem 5.5 *The problem "given a finite propositional logic program from class $\mathcal{E}_k$, decide whether P has a complete ultimate stable model" is* NP*-complete.*

Theorem 5.6 *The problem "given a finite propositional logic program from class $\mathcal{E}_k$, compute the ultimate well-founded fixpoint of P" is in* P.

We will now prove these results. If P is a finite propositional program, then it follows directly from the definition of the ultimate Kripke-Kleene fixpoint of T_P (that is, the ultimate Kripke-Kleene model of P) that it can be computed by means of polynomially many (in the size of P) evaluations of the operator $U_P(I, J)$, where $I \subseteq J$ are interpretations, with all other computational tasks taking only polynomial amount of time.

Let us also note that I is a complete ultimate stable model of P if and only if $I = lfp(U_P(\cdot, I))$. Thus, to verify whether I is a complete ultimate stable model, it is enough to iterate the operator $lfp(U_P(\cdot, I))$ starting with the empty interpretation. The number of iterations needed to reach the least fixpoint is again polynomial in the size of P with all other needed tasks taking polynomial time only. A similar discussion shows that the ultimate well-founded model of P can be computed by means of polynomially many evaluations of

the form $U_P(I,J)$.

It follows that evaluating $U_P(I,J)$, where $I \subseteq J$, is at the heart of computing the ultimate Kripke-Kleene, well-founded and complete stable models of a program P. Hence, we will now focus on this task.

Let P be a logic program and let p be an atom in P. For every rule $r \in P$ such that p is the head of r, we define B_r to be the conjunction of all literals in the body of r. For every atom p, we denote by $B_P(p)$ the disjunction of all formulas B_r, where r ranges over all rules in P with the head p. When p is the head of no rule in P then we set $B_P(r) = \bot$ (empty disjunction).

Every logic program P has a *normal* representation. It is the collection of rules $p \leftarrow B_P(p)$, where p ranges over all atoms of P. The definition of the operator T_P extends, in a straightforward way, to the case when P is given in its normal form defined above. Moreover, if P is a logic program and Q is its normal representation, $T_P = T_Q$. Thus, in the remainder of this section, without loss of generality we will assume that programs are given by means of their normal representations.

Let us recall that

$$U^1_P(I,J) = glb(T_P([I,J])) = \bigcap_{I \subseteq K \subseteq J} T_P(K)$$

and

$$U^2_P(I,J) = lub(T_P([I,J])) = \bigcup_{I \subseteq K \subseteq J} T_P(K).$$

Let I and J be two interpretations such that $I \subseteq J$. We define the *reduct* $P_{I,J}$ of P to be the program obtained from P by substituting in each body formula $B_P(p)$, any atom r by f if $r \notin J$ and any atom r by t if $r \in I$. Note that all body atoms of $P_{I,J}$ is an element of $J \setminus I$.

We have the following simple properties. An atom p of P belongs to $U^1_P(I,J)$ if and only if for every interpretation $K \in [\emptyset, J \setminus I]$, the formula $B_{P_{I,J}}(p)$ is true in K (or, equivalently, if and only if the formula $B_{P_{I,J}}(p)$ is a tautology). An atom p of P belongs to $U^2_P(I,J)$ if and only if for some interpretation $K \in [\emptyset, J \setminus I]$, the formula $B_{P_{I,J}}(p)$ is true in K (or, equivalently, if and only if the formula $B_{P_{I,J}}(p)$ is satisfiable).

From the second property it follows that computing $U^2_P(I,J)$ is easy — it can be accomplished in polynomial time (in the size of P). Indeed, since $B_{P_{I,J}}(p)$ is a DNF formula, its satisfiability can be decided in polynomial time and the claim follows. Thus, from now on we will focus on the task of computing $U^1_P(I,J)$.

The problem to decide whether a DNF formula is a tautology is co-NP-complete. Thus, the problem to compute the ultimate Kripke-Kleene and well-founded models of a program P is in the class Δ^P_2. Consequently, Theorem 5.4 follows.

It also follows that checking whether for an interpretation J, $J = lfp(U^1_P(\cdot, J))$ is in Δ^P_2. Hence, the problem to decide whether a program has a complete ultimate stable fixpoint is in the class Σ^P_2.

We will now show the Σ^P_2-hardness of the problem of existence of a complete ultimate stable model of a program P. Let φ be a propositional formula and let I be an interpretation (a set of atoms). We recall that the following problem is Σ^P_2-complete: Given a DNF formula φ over variables $x_1, \ldots, x_m$, $y_1, \ldots, y_n$, decide whether there is a truth assignment $I \subseteq \{x_1, \ldots, x_m\}$ such that φ_I is a tautology, where φ_I is the formula obtained by replacing in φ all occurrences of atoms from I with t, and by replacing all occurrences of atoms from $\{x_1, \ldots, x_m\} \setminus I$ with f.

We will reduce this problem to our problem. For each x_i, $i = 1, \ldots, m$, in φ we introduce a new variable x'_i. We also introduce two new atoms p and q. By φ' we denote the formula obtained from φ by replacing literals $\neg x_i$ in the disjuncts of φ with new atoms x'_i. We define a program $P(\varphi)$ to consist of the following clauses:

1. $x_i \leftarrow \mathbf{not}(x'_i)$ and $x'_i \leftarrow \mathbf{not}(x_i)$, for every $i = 1, \ldots, m$
2. $y_i \leftarrow \varphi'$, for every $i = 1, \ldots, n$
3. $p \leftarrow \varphi'$
4. $q \leftarrow \mathbf{not}(p), \mathbf{not}(q)$.

We will show that there is $I \subseteq \{x_1, \ldots, x_m\}$ such that φ_I is a tautology if and only if $P(\varphi)$ has an ultimate complete stable model.

It is easy to see the that the following properties hold for every fixpoint M of $T_{P(\varphi)}$:

1. q is false in M (if q is true in M, $T_{P(\varphi)}$ does not derive q);
2. p is true in M (otherwise $T_{P(\varphi)}$ derives q);
3. $y_1, .., y_n$ are true in M (since their rules have the same bodies as p);
4. for each x_i, either x_i or x'_i is true in M.

For a subset $I \subseteq \{x_1, \ldots, x_m\}$, let us define $\overline{I} = I \cup \{x'_i : x_i \notin I\}$. It follows from the properties listed above that for each fixpoint M of $T_{P(\varphi)}$ and, a fortiori, if M is a complete ultimate stable model of $P(\varphi)$, there exists an I such that

$$M = \overline{I} \cup \{p, y_1, \ldots, y_n\}.$$

Thus, it suffices to show that if $I \subseteq \{x_1, \ldots, x_m\}$ then $M = \overline{I} \cup \{p, y_1, \ldots, y_n\}$ is a complete ultimate stable model of $P(\varphi)$ if and only if φ_I is a tautology.

It is easy to verify that for every set $M = \overline{I} \cup \{p, y_1, \ldots, y_n\}$ and for every $J \subseteq M$, $U^1(J, M)$ satisfies the following properties:

1. $U^1_{P(\varphi)}(J, M) \cap \{x_1, .., x_n, x'_1, .., x'_n\} = \overline{I}$
2. $U^1_{P(\varphi)}(J, M) \cap \{y_1, .., y_n, p, q\}$ is either $\emptyset$ or $\{y_1, .., y_n, p\}$, since bodies of rules of $y_1, .., y_n, p$ are identical.

Thus, we find that $U^1_{P(\varphi)}(J, M)$ is either $\overline{I}$ or M and, consequently, $U^1_{P(\varphi)}(\cdot, M)$ has a least fixpoint, which is either $\overline{I}$ or M. Hence $M = \overline{I} \cup \{p, y_1, \ldots, y_n\}$ is a complete ultimate stable model of $P(\varphi)$ if and only if $\overline{I}$ is not a fixpoint of $U^1_{P(\varphi)}(\cdot, M)$, that is if $U^1_{P(\varphi)}(\overline{I}, M) = M$. Consequently, all we need to prove is that $p \in U^1_{P(\varphi)}(\overline{I}, M)$ if and only if φ_I is a tautology.

Let us recall that $p \in U^1_{P(\varphi)}(\overline{I}, M)$ if and only if for every interpretation $K \in [\emptyset, M \setminus \overline{I}]$, the formula $B_{P(\varphi)_{T,M}}(p)$ is true in K, that is, if and only if the formula $B_{P(\varphi)_{T,M}}(p)$ is a tautology. Let us observe that $B_{P(\varphi)}(p) = \varphi'$. Thus, it is easy to see that $B_{P(\varphi)_{T,M}}(p)$ is logically equivalent to φ_I. Consequently, the claim and Theorem 5.3 follows.

The problems of interest restricted to programs from the class $\mathcal{E}_k$ become easier. Let us recall that the decision whether an atom $p \in At(P)$ belongs to $U^1_P(I, J)$ boils down to the decision whether the formula $B_{P_{I,J}}(p)$ is a tautology. If P is in the class $\mathcal{E}_k$, this question can be resolved in polynomial time. Thus, the ultimate Kripke-Kleene and the well-founded models for programs in $\mathcal{E}_k$ can be computed in polynomial time. Thus, Theorem 5.6 follows.

Similarly, it takes only polynomial time to verify whether an interpretation I satisfies $I = lfp(U^1_P(\cdot, I))$. Thus, the problem to decide whether a program from $\mathcal{E}_k$ has a complete ultimate stable model is in NP. To prove completeness, we observe that for purely negative programs:

1. there is no difference between complete stable fixpoints and complete ultimate stable fixpoints
2. purely negative programs are in $\mathcal{E}_k$
3. the problem of existence of complete stable fixpoints for purely negative programs is NP-complete.

Thus, Theorem 5.5 follows.

6 CONCLUSIONS AND DISCUSSION

We extended our algebraic framework [DMT00a, DMT00b] for studying semantics of nonmonotonic reasoning systems. The main contribution of this paper is the notion of an ultimate approximation. We argue that the Kripke-Kleene, well-founded and stable fixpoints of the ultimate approximation of an operator O can be regarded as the Kripke-Kleene, well-founded and stable fixpoints of the operator O itself. In earlier approaches, to study fixpoints of an operator O one needed to *select* an appropriate approximation operator. There were, however, no principled, algebraic ways to do so. In the present paper, we find a *distinguished* element in the space of all approximations and propose this particular approximation (ultimate approximation) to study fixpoints of O.

A striking feature of our approach is the ease with which it can be applied in any context where semantics emerge as fixpoints of operators. We applied this approach here in the context of logic programming and obtained a family of new semantics for logic programs: the ultimate Kripke-Kleene, the ultimate well-founded and the ultimate stable-model semantics. These semantics are well motivated and have attractive properties. First, they are preserved when we modify the program, as long as the 2-valued provability operator stays the same (the property that does not hold in general for standard semantics). Second, the ultimate Kripke-Kleene and the well-founded semantics are stronger (in general) than their standard counterparts, yet approximate the collection of all fixpoints of O and the collection of all stable fixpoints of O, respectively. The disadvantage is that their complexity is higher. But, as we noticed, for large classes of programs there is actually no loss in efficiency of computing ultimate semantics.

This approach can also be applied to default and autoepistemic logics and results in new semantics with appealing epistemological features[2]. It was also recently used to define a precise semantics for logic programs with aggregates [DPB01].

We end this discussion with comments on a possible broader role of the approximation theory. One common concern when designing semantics of nonmonotonic logics is to avoid models justified by ungrounded or self-supporting (circular) arguments. The well-founded fixpoints (semantics) avoid such arguments. Groundedness is also a fundamental feature of induc-

[2]We will include a more extensive discussion of these applications in the journal version of the paper.

tion, a constructive way in which humans specify concepts both in commonsense reasoning settings and in formal considerations. In its simplest form induction relies only on positive information. In general, however, it may make references to negative information, too. In either form it is a nonmonotonic specification mechanism. As argued in [Den98], the well-founded semantics generalizes existing formalizations of induction (for instance, positive induction and iterated induction .

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants No. 9874764 and 0097278. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[AvE82] K.R. Apt and M.H. van Emden. Contributions to the theory of logic programming. *Journal of the ACM*, 29(3):841–862, 1982.

[Cla78] K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and data bases*, pages 293–322. Plenum Press, New York-London, 1978.

[Den98] M. Denecker. The well-founded semantics is the principle of inductive definition. In J. Dix, L. Fariñas del Cerro, and U. Furbach, eds., *Logics in Artificial Intelligence*, LNAI, volume 1489, Springer-Verlag 1998.

[DMT00a] M. Denecker, V. Marek, and M. Truszczyński. Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In J. Minker, editor, *Logic-Based Artificial Intelligence*, pages 127–144. Kluwer Academic Publishers, 2000.

[DMT00b] 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)*, pages 74 – 84. Morgan Kaufmann Publishers, 2000.

[DPB01] M. Denecker, N. Pelov, and M. Bruynooghe. Well-founded and stable semantics for logic programs with aggregates. In *Proceedings of ICLP-01*, LNCS 2237, Springer-Verlag, 2001.

[Fit85] M. C. Fitting. A Kripke-Kleene semantics for logic programs. *Journal of Logic Programming*, 2(4):295–312, 1985.

[Fit01] M. C. Fitting. Fixpoint semantics for logic programming – a survey. *Theoretical Computer Science*, 2002. To appear.

[Gin88] M.L. Ginsberg. Multivalued logics: a uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.

[GL88] M. Gelfond and V. Lifschitz. The stable semantics for logic programs. In R. Kowalski and K. Bowen, editors, *Proceedings of the 5th International Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[Kun87] K. Kunen. Negation in logic programming. *Journal of Logic Programming*, 4(4):289–308, 1987.

[Mar76] G. Markowsky. Chain-complete posets and directed sets with applications. *Algebra Universalis*, 6(1):53–68, 1976.

[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.

[Prz90] T.C. Przymusinski. The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae*, 13(4):445–464, 1990.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.

[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.

Description Logics

Adding Numbers to the $\mathcal{SHIQ}$ Description Logic—First Results

Carsten Lutz
LuFG Theoretical Computer Science, RWTH Aachen
Ahornstr. 55, 52074 Aachen, Germany

Abstract

Recently, the Description Logic (DL) $\mathcal{SHIQ}$ has found a large number of applications. This success is due to the fact that $\mathcal{SHIQ}$ combines a rich expressivity with efficient reasoning. One weakness of $\mathcal{SHIQ}$, however, limits its usability in several application areas: numerical knowledge such as knowledge about the age, weight, or temperature of real-world entities cannot be adequately represented. In this paper, we present $\mathbb{Q}$-$\mathcal{SHIQ}$, an extension of $\mathcal{SHIQ}$ that aims at closing this gap, and show that reasoning with the extended DL is ExpTime-complete.

1 Motivation

Description Logics (DLs) are a family of knowledge representation formalisms, which are—apart from their classical application in KR—nowadays used in various application areas such as reasoning about entity relationship (ER) diagrams and providing a formal basis for the so-called semantic web [4; 6]. One of the most influential DLs proposed during the last years is the $\mathcal{SHIQ}$ Description Logic, whose success is based mainly on the following two facts: first, $\mathcal{SHIQ}$ is a very expressive DL providing for, e.g., transitive roles, inverse roles, and number restrictions, but its reasoning problems are nevertheless decidable in ExpTime [13; 22]. Second, $\mathcal{SHIQ}$ has been implemented in efficient DL systems such as FaCT and RACER, which can, despite the high worst-case complexity of reasoning with $\mathcal{SHIQ}$, deal surprisingly well even with huge knowledge bases [10; 7].

Although, as we just argued, $\mathcal{SHIQ}$'s expressive power is one of the main reasons for its success, there is still room for improvement. In particular, $\mathcal{SHIQ}$ cannot adequately represent numerical knowledge such as knowledge about the age, weight, or temperature of real-world entities, which, as we will later discuss in more detail, is crucial for many important applications [2; 12; 6; 17]. In this paper, we extend $\mathcal{SHIQ}$ with a set of concept constructors that belong to the so-called concrete domain family of constructors and allow a straightforward representation of numerical knowledge. Let us view a concrete example of knowledge representation with the resulting DL, which is called $\mathbb{Q}$-$\mathcal{SHIQ}$: the concept

$$\mathsf{Grandfather} \sqcap \exists \mathsf{age}.=_{91} \sqcap (\geqslant 20\ \mathsf{relatives}\ \mathsf{Human}) \sqcap \forall \mathsf{relatives}\ \mathsf{age}, \mathsf{age}.<$$

describes Grandfathers who are 91 years old, have at least 20 relatives (such constraints are called "qualified number restrictions"), and are older than all of these relatives. Note that we can refer to rational numbers such as "91" and also compare numbers using predicates such as "$<$". We argue that the additional expressivity provided by $\mathbb{Q}$-$\mathcal{SHIQ}$ is rather useful in many application areas. Let us briefly review three examples:

(1) As described in [5; 4], reasoning about ER diagrams is an important application area of Description Logics. One shortcoming of the standard way to encode ER diagrams, which is to use a fragment of the $\mathcal{SHIQ}$ Description Logic, can be described as follows: ER diagrams make use of so-called attributes to represent non-relational data such as numbers and strings to be stored in the database. If $\mathcal{SHIQ}$ is used for representing ER diagrams, constraints concerning the values of attributes cannot be expressed. To give a simple example, if there exists an entity Employee having two attributes Birthday and Employment-date, then it cannot be expressed that employees should be born before they are hired. If $\mathbb{Q}$-$\mathcal{SHIQ}$ is used for representing ER diagrams, such *numerical* data constraints on attributes can easily be handled. This topic is discussed in more detail in [19].

(2) In [17; 16], the Description Logic $\mathcal{TDL}$ is motivated as a valuable tool for the representation of temporal conceptual knowledge. $\mathcal{TDL}$ can be obtained from the well-known DL $\mathcal{ALC}$ [20] by adding general TBoxes and concrete domain style concept constructors that allow to represent relations between rational numbers such as "=" and "<". Indeed, it is not hard to see that $\mathcal{TDL}$ is a proper fragment of $\mathbb{Q}$-$\mathcal{SHIQ}$. Thus, $\mathbb{Q}$-$\mathcal{SHIQ}$ is also well-suited for reasoning about temporal conceptual knowledge as described in [17; 16]. Moreover, $\mathbb{Q}$-$\mathcal{SHIQ}$ significantly extends the expressive power provided by $\mathcal{TDL}$, even in the temporal/numerical component of the logic. For example, if $\mathbb{Q}$-$\mathcal{SHIQ}$ is used for temporal reasoning, then one can refer to *concrete* time points and time intervals such as 4 or $[1, 12]$. This is not possible in $\mathcal{TDL}$.

(3) A rapidly developing application area of DLs is their use as an ontology language for the semantic web [6]. As noted in [6; 9], the representation of "concrete datatypes" such as numbers is an important task in this context. However, in DLs such as OIL and DAML+OIL, which have been proposed in this application area, appropriate expressivity is either not provided or not taken into account for reasoning, which is done by a translation into $\mathcal{SHIQ}$ or related DLs. In [12], Horrocks and Sattler propose to extend $\mathcal{SHOQ}$, a close relative of $\mathcal{SHIQ}$, with so-called unary concrete domains in order to integrate concrete datatypes. However, this solution is not really satisfying since, as is explained in more detail in [16], unary concrete domains are of very limited expressivity. If $\mathbb{Q}$-$\mathcal{SHIQ}$ is used as the target logic in translations of OIL and DAML+OIL, a rather powerful means for describing *numerical* concrete datatypes becomes available.

As the main result of this paper, we prove reasoning with $\mathbb{Q}$-$\mathcal{SHIQ}$ to be decidable in ExpTime by devising an automata-based decision procedure. Thus, $\mathbb{Q}$-$\mathcal{SHIQ}$ sensibly enhances the expressive power of $\mathcal{SHIQ}$ without increasing the worst-case complexity of reasoning. This paper is accompanied by a technical report that contains more details and full proofs [14].

2 Syntax and Semantics

In this section, we introduce the Description Logic $\mathbb{Q}$-$\mathcal{SHIQ}$ in detail. We first give the syntax and semantics of $\mathbb{Q}$-$\mathcal{SHIQ}$-roles, then introduce some useful abbreviations, and finally define syntax and semantics of $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts.

Definition 1. Let N_{rR}, N_{tR}, and N_{aF} be countably infinite and mutually disjoint sets of *regular role names*, *transitive role names*, and *abstract features*, respectively. Moreover, let $\mathsf{N}_{\mathsf{R}} = \mathsf{N}_{\mathsf{rR}} \uplus \mathsf{N}_{\mathsf{tR}} \uplus \mathsf{N}_{\mathsf{aF}}$. The set of $\mathbb{Q}$-$\mathcal{SHIQ}$-*roles* ROL is $\mathsf{N}_{\mathsf{R}} \cup \{R^- \mid R \in \mathsf{N}_{\mathsf{R}}\}$. A *role inclusion* is of the form $R \sqsubseteq S$, for $R, S \in \mathsf{ROL}$. A *role hierarchy* is a set of role inclusions.

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$, called the *domain* of $\mathcal{I}$, and a function $\cdot^{\mathcal{I}}$ which maps every role $R \in \mathsf{ROL}$ to a subset $R^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that, for $R \in \mathsf{N}_{\mathsf{R}}$, $S \in \mathsf{N}_{\mathsf{tR}}$, and $f \in \mathsf{N}_{\mathsf{aF}}$, we have

- $(x, y) \in R^{\mathcal{I}}$ iff $(y, x) \in R^{-\mathcal{I}}$,
- if $(x, y) \in S^{\mathcal{I}}$ and $(y, z) \in S^{\mathcal{I}}$, then $(x, z) \in S^{\mathcal{I}}$,
- $f^{\mathcal{I}}$ is functional.

An interpretation $\mathcal{I}$ is a *model* of a role hierarchy $\mathcal{R}$ iff $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ for each $R \sqsubseteq S \in \mathcal{R}$.

We introduce some notation to make the following considerations easier:

(1) The function Inv yields the inverse of a role. More precisely, for $R \in \mathsf{ROL}$, we set

$$\mathsf{Inv}(R) := \begin{cases} R^- & \text{if } R \text{ is a role name,} \\ S & \text{if } R = S^- \text{ for a role name } S. \end{cases}$$

(2) Since set inclusion is transitive and $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ implies $\mathsf{Inv}(R)^{\mathcal{I}} \subseteq \mathsf{Inv}(S)^{\mathcal{I}}$, for a role hierarchy $\mathcal{R}$, we introduce $\sqsubseteq_{\mathcal{R}}$ as the reflexive-transitive closure of $\mathcal{R} \cup \{\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \mid R \sqsubseteq S \in \mathcal{R}\}$.

(3) We call a role $R \in \mathsf{ROL}$ *transitive* with respect to a role hierarchy $\mathcal{R}$ iff R is interpreted in a transitive relation in every model of $\mathcal{R}$. It is not hard to see that this is the case iff the following predicate evaluates to true:

$$\mathsf{Trans}_{\mathcal{R}}(R) := \begin{cases} \mathsf{true} & \text{if there exists a role } S \in \mathsf{N}_{\mathsf{tR}} \\ & \quad \text{s.t. } S' \sqsubseteq_{\mathcal{R}} R \text{ and } R \sqsubseteq_{\mathcal{R}} S'' \\ & \quad \text{for some } S', S'' \in \{S, \mathsf{Inv}(S)\} \\ \mathsf{false} & \text{otherwise.} \end{cases}$$

(4) A role $R \in \mathsf{ROL}$ is called *simple* with respect to a role hierarchy $\mathcal{R}$ iff $\mathsf{Trans}_{\mathcal{R}}(S)$ does not hold for any $S \in \mathsf{ROL}$ with $S \sqsubseteq R$.

For both "$\sqsubseteq_{\mathcal{R}}$" and $\mathsf{Trans}_{\mathcal{R}}$, we omit the index if clear from the context. Note that no transitive role is simple since $\sqsubseteq$ is defined as the *reflexive*-transitive closure. For the same reason, we have $\mathsf{Trans}(R)$ for all $R \in \mathsf{N}_{\mathsf{tR}}$. However, roles must obviously not be in N_{tR} in order to be transitive. For example, if $R \in \mathsf{N}_{\mathsf{tR}}$, then R^- is also transitive. Similarly, if $S \in \mathsf{N}_{\mathsf{tR}}$, $R \notin \mathsf{N}_{\mathsf{tR}}$, $S^- \sqsubseteq R$, $R \sqsubseteq S^-$, then R is transitive.

$$
\begin{array}{rcl}
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}}, \quad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}, \quad \neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, \\
(\exists R.C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid \text{There is some } y \in \Delta^{\mathcal{I}} \text{ with } (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}, \\
(\forall R.C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid \text{For all } y \in \Delta^{\mathcal{I}}, \text{ if } (x,y) \in R^{\mathcal{I}}, \text{ then } y \in C^{\mathcal{I}}\}, \\
(\leqslant n\ R\ C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \leqslant n\}, \\
(\geqslant n\ R\ C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \geqslant n\}, \\
(\exists U_1, U_2.P)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid \text{There are } q_1 \in U_1^{\mathcal{I}} \text{ and } q_2 \in U_2^{\mathcal{I}} \text{ with } q_1\, P\, q_2\} \\
(\forall U_1, U_2.P)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid \text{For all } q_1 \in U_1^{\mathcal{I}} \text{ and } q_2 \in U_2^{\mathcal{I}}, \text{ we have } q_1\, P\, q_2\} \\
(\exists g.P_q)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid g^{\mathcal{I}}(x) \text{ is defined and } g^{\mathcal{I}}(x)\ P\ q\}
\end{array}
$$

Figure 1: $\mathbb{Q}$-$\mathcal{SHIQ}$ concept semantics.

We are now ready to define $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts and their semantics.

Definition 2. Let $\mathsf{N_C}$ and $\mathsf{N_{cF}}$ be countably infinite sets of *concept names* and *concrete features*, respectively, such that $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_{cF}}$ are mutually disjoint. A *path* is a sequence $R_1 \cdots R_k g$ consisting of roles $R_1, \ldots, R_k \in \mathsf{ROL}$ and a concrete feature $g \in \mathsf{N_{cF}}$. A path $R_1 \cdots R_k g$ in which $R_1, \ldots, R_k$ are abstract features (i.e., from $\mathsf{N_{aF}}$) is called *feature path*. The set of $\mathbb{Q}$-$\mathcal{SHIQ}$-*concepts* is the smallest set such that

1. every concept name $C \in \mathsf{N_C}$ is a concept,
2. if C and D are concepts and $R \in \mathsf{ROL}$, then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall R.C$, and $\exists R.C$ are concepts,
3. if C is a concept, $R \in \mathsf{ROL}$ is simple, and $n \in \mathbb{N}$, then $(\leqslant n\ R\ C)$ and $(\geqslant n\ R\ C)$ are concepts,
4. if u_1 and u_2 are feature paths and P is a predicate from the set $\{<, \leq, =, \neq, \geq, >\}$, then $\exists u_1, u_2.P$ and $\forall u_1, u_2.P$ are concepts,
5. if $R \in \mathsf{ROL}$ is simple, g_1 and g_2 are concrete features, and $P \in \{<, \leq, =, \neq, \geq, >\}$, then $\exists R g_1, g_2.P$ and $\forall R g_1, g_2.P$ are concepts, and
6. if g is a concrete feature, $P \in \{<, \leq, =, \neq, \geq, >\}$, and $q \in \mathbb{Q}$, then $\exists g.P_q$ is a concept.

We use $\top$ as an abbreviation for $A \sqcup \neg A$ (for some fixed $A \in \mathsf{N_C}$). The interpretation function $\cdot^{\mathcal{I}}$ of interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ maps, additionally, every concept C to a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and every concrete feature g to a partial function $g^{\mathcal{I}}$ from $\Delta^{\mathcal{I}}$ to the set of rational numbers $\mathbb{Q}$ such that the equations in Figure 1 are satisfied, where U_1 and U_2 denote paths, $\sharp S$ denotes the cardinality of the set S, and, for every path $U = R_1 \cdots R_k g$, $U^{\mathcal{I}}$ is defined as

$$
\begin{array}{l}
\{(x,q) \subseteq \Delta_{\mathcal{I}} \times \mathbb{Q} \mid\ \exists y_1, \ldots, y_{k+1} : x = y_1, \\
\qquad (y_i, y_{i+1}) \in R_i^{\mathcal{I}} \text{ for } 1 \leq i \leq k, \text{ and } g^{\mathcal{I}}(y_{k+1}) = q\}.
\end{array}
$$

An interpretation $\mathcal{I}$ is a *model* of a concept C iff $C^{\mathcal{I}} \neq \emptyset$. C is called *satisfiable with respect to a role hierarchy* $\mathcal{R}$ iff there exists a model of C and $\mathcal{R}$. A concept D *subsumes* a concept C with respect to $\mathcal{R}$ (written $C \sqsubseteq_{\mathcal{R}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for each model $\mathcal{I}$ of $\mathcal{R}$.

Throughout this paper, we denote concept names by A and B, concepts by C, D, and E, roles by P, R, and S, abstract features by f, concrete features by g, paths by U, feature paths by u, and predicates by P.

In the following sections, we show that $\mathbb{Q}$-$\mathcal{SHIQ}$-concept satisfiability is decidable in deterministic exponential time. This also yields decidability and an ExpTime upper complexity bound for concept subsumption: we have $C \sqsubseteq_{\mathcal{R}} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{R}$.

Most modern Description Logics do not only consist of a concept language but also provide for a TBox component. Formally, a TBox is a finite set of concept equations $C \doteq D$, where C and D are concepts, and an interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ iff it satisfies $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all $(C \doteq D) \in \mathcal{T}$. In the presence of TBoxes, one is usually interested in the satisfiability of concepts w.r.t. TBoxes and role hierarchies, i.e., in whether there exists a model $\mathcal{I}$ of C, $\mathcal{T}$, and $\mathcal{R}$. However, as shown in [11], in the presence of role hierarchies and transitive roles, it is possible to polynomially reduce concept satisfiability w.r.t. TBoxes and role hierarchies to concept satisfiability w.r.t. role hierarchies, only. Hence, we do not explicitly consider TBoxes in what follows.

Let us discuss the $\mathbb{Q}$-$\mathcal{SHIQ}$-concept language in some more detail. Since exhaustive information on $\mathcal{SHIQ}$ can be found in, e.g., [13], we concentrate on the additional concept constructors $\exists U_1, U_2.P$, $\forall U_1, U_2.P$, and $\exists g.P_q$, which, as has already been noted, are often called "concrete domain constructors". Concrete domains have been introduced by Baader and Hanschke as a means for representing "concrete knowledge" such as knowledge about numbers, strings, and spatial extensions [1]. More precisely, Baader and Hanschke extend the basic propositionally closed DL $\mathcal{ALC}$ with

concrete domains, where a concrete domain $\mathcal{D}$ is comprised of a set called the domain and a set of predicates with a fixed extension on this domain. It is important to note that Baader and Hanschke do not commit themselves to a particular concrete domain, but rather view the concrete domain $\mathcal{D}$ as a parameter to their logic, which they call $\mathcal{ALC}(\mathcal{D})$. From the concrete domain perspective, $\mathbb{Q}$-$\mathcal{SHIQ}$ can be viewed as being equipped with one particular concrete domain, whose domain are the rationals and which is equipped with binary predicates $<, \leq, =, \neq, \geq, >$ and with (infinitely many) unary predicates P_q, where $q \in \mathbb{Q}$ and $P \in \{<, \leq, =, \neq, \geq, >\}$.

The paths U_1 and U_2 that may appear inside $\mathbb{Q}$-$\mathcal{SHIQ}$'s binary concrete domain constructors $\exists U_1, U_2.P$ and $\forall U_1, U_2.P$ are of a rather special form: either (i) U_1 and U_2 are feature paths or (ii) U_1 has the form Rg_1 and U_2 has the form g_2. Let us illustrate the expressive power of these two variants of the same constructors: using Variant (i), we can, e.g., describe people whose mother's spouse earns more than their father (we use parentheses for better readability):

$$\exists(\mathsf{mother\ spouse\ wage}), (\mathsf{father\ wage}).>$$

The example illustrates the main advantage of Variant (i): we can talk about *sequences* of features. This variant of $\mathbb{Q}$-$\mathcal{SHIQ}$'s binary concrete domain constructors are precisely the concrete domain constructors offered by $\mathcal{ALC}(\mathcal{D})$ and the temporal DL $\mathcal{TDL}$ mentioned in the introduction. The main disadvantage of Variant (i) is that, inside paths, we may only use abstract features but no roles from N_{fR}. For example, if we want to describe people having an older neighbor by the concept

$$\exists(\mathsf{neighbor}, \mathsf{age}), (\mathsf{age}).>,$$

then "neighbor" should clearly be from N_{fR} rather than from N_{aF}, since otherwise we would enforce that the described persons have at most a single neighbor. Therefore, we need Variant (ii) of the binary concrete domain constructors to define this concept. Note that Variant (ii) is neither provided by $\mathcal{ALC}(\mathcal{D})$ nor by $\mathcal{TDL}$, but rather is a restricted version of the concrete domain constructors defined in [8].

It is not hard to see that we could also have admitted variants $\exists g_1, Rg_2.P$ and $\forall g_1, Rg_2.P$ of the binary concrete constructors since this variant is just syntactic sugar: $\exists g_1, Rg_2.P$ is equivalent to $\exists Rg_2, g_1.\widetilde{P}$ and $\forall g_1, Rg_2.P$ is equivalent to $\forall Rg_2, g_1.\widetilde{P}$, where $\widetilde{P}$ denotes the inverse of the predicate P—for example, "$\widetilde{<}$" is "$>$" and "$\widetilde{=}$" is "$=$". Obviously, the most general approach would be to allow arbitrary paths inside the binary concrete domain constructors.[1] The resulting logic, however, cannot easily be handled by the ExpTime decision procedure for concept satisfiability presented in the remainder of this paper.

Only simple roles are allowed in Variant (ii) of the binary concrete domain constructors. Similarly, roles used inside number restrictions are also required to be simple. As proved in [13], the latter restriction is crucial since admitting non-simple roles inside number restrictions yields undecidable reasoning problems. Non-simple roles inside the binary concrete domain constructors cannot be handled by the ExpTime decision procedure presented in this paper. However, it is as of now unknown whether admitting them yields undecidability of reasoning.

As a last comment concerning the binary concrete domain constructors, note that there exist existential and universal versions of the binary concrete domain constructors but only an existential version of the unary concrete domain constructor. It is not hard to see that we could also have admitted a universal version since $\forall g.P_q$ (with the obvious semantics) is clearly equivalent to $\forall g, g.\neq \sqcup \exists g.P_q$, where $\forall g, g.\neq$ simply expresses that there exists no successor for the concrete feature g. Similarly, the universal version of Variant (i) of the binary concrete domain constructors can be expressed in terms of the existential version of Variant (i) of this constructor. This does, however, not hold for Variant (ii) of the binary concrete domain constructors since it accepts non-functional roles as arguments. For this reason, we have chosen to include universal versions of both Variant (i) and (ii) for uniformity.

It may look strange at first sight that $\mathbb{Q}$-$\mathcal{SHIQ}$ provides for both abstract features and number restrictions since, as is well-known, number restrictions, transitive roles, and role hierarchies can be used to enforce that a role R_f from N_{fR} is interpreted functionally: just use the concept $\forall R.(\leqslant 1\ R_f\ \top)$, where $R \in \mathsf{N}_{\mathsf{tR}}$, and employ the role hierarchy to ensure that $S \sqsubseteq R$ for every "relevant" role S (i.e. for the roles occurring in the concept and role hierarchy whose satisfiability is to be decided). The reason for this redundancy is that number restrictions are, in principle, a strictly more general means of expressivity than abstract features, but having abstract features explicitly available allows for a straightforward definition of Variant (i) of the concrete domain constructors.

[1] Admitting arbitrary paths inside the *unary* concrete domain constructor is not an issue since the concept $\exists R_1 \cdots R_k g.P_q$ (with the obvious semantics) can be written as $\exists R_1. \cdots .\exists R_k.\exists g.P_q$.

3 Preliminaries

Decidability and the ExpTime upper complexity bound for $\mathbb{Q}$-$\mathcal{SHIQ}$-concept satisfiability is established by devising an automata-based decision procedure. The general idea behind this procedure is to define, for a given concept C and role hierarchy $\mathcal{R}$, a looping tree-automaton $\mathcal{A}_{C,\mathcal{R}}$ that accepts exactly the so-called Hintikka-trees for C and $\mathcal{R}$. These Hintikka-trees are abstractions of models of C and $\mathcal{R}$, i.e., C and $\mathcal{R}$ have a model if and only if C and $\mathcal{R}$ have a Hintikka-tree. The obvious advantage of Hintikka-trees over models is that they are trees and thus amenable to tree automata techniques. Once the automaton $\mathcal{A}_{C,\mathcal{R}}$ is defined, it remains to apply the standard emptiness test for tree automata: clearly, the language accepted by the constructed automaton is empty iff C is satisfiable w.r.t. $\mathcal{R}$.

In this section, we introduce the basic notions underlying the decision procedure sketched above. We start with developing a useful normal form (called *path normal form*) for $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts, and then introduce looping tree-automata. Finally, we define constraint graphs, which will play an important role in representing the "numerical part" of $\mathbb{Q}$-$\mathcal{SHIQ}$-interpretations in Hintikka-trees.

3.1 Normal Forms

We start with formulating a property of role hierarchies that we will generally assume to be satisfied in what follows:

> A role hierarchy $\mathcal{R}$ is called *admissible* iff all $f \in \mathsf{N_{aF}}$ are simple w.r.t. $\mathcal{R}$.

Demanding admissibility of role hierarchies is closely related to requiring roles R that appear inside number restrictions $(\leqslant n\ R\ C)$ and $(\geqslant n\ R\ C)$ to be simple: since abstract features are interpreted in functional relations, they are "inherently number restricted", i.e., for each $f \in \mathsf{N_{aF}}$, $(\leqslant 1\ f\ \top)$ is satisfied by every domain element in every interpretation. However, it seems that, in contrast to admitting arbitrary roles inside number restrictions, dropping admissibility of role hierarchies does not seem to lead to undecidability of reasoning. Indeed, we claim that the decision procedure presented in this paper can, in principle, be extended to also deal with non-admissible role hierarchies. We nevertheless restrict ourselves to admissible role hierarchies since (i) this eliminates several case distinctions in the proofs, and (ii) we agree with Horrocks and Sattler [11] who argue that non-simple features are rather unnatural: if $f \in \mathsf{N_{aF}}$ is non-simple, then there exists a role $R \in \mathsf{N_R}$ such that $\mathsf{Trans}(R)$ and $R \sqsubseteq f$. Hence, R is both functional and transitive which produces strange effects: for any interpretation $\mathcal{I}$, $R^{\mathcal{I}}$ may not contain any acyclic paths of length greater 1. Hence, the concept $\exists R.\exists R.\top$ is satisfiable only in models that contain either (i) a domain element a which is its own R-successor or (ii) two domain elements a and b, where b is R-successor of a and of itself (the same holds for the concept $\exists R.\exists R.\exists R.\top$). To avoid such effects, which do not seem to promote writing understandable knowledge bases, we generally require role hierarchies to be admissible.

Let us now turn our attention towards the path normal form for $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts, which was first described in [17] in the context of the Description Logic $\mathcal{TDL}$.

Definition 3. A $\mathbb{Q}$-$\mathcal{SHIQ}$-concept C is in *negation normal form* (NNF) if negation occurs only in front of concept names. Moreover, C is in *path normal form* (PNF) iff it is in NNF and, for all subconcepts $\exists U_1, U_2.P$ and $\forall U_1, U_2.P$ of C, we have either

1. $U_1 = g_1$ and $U_2 = g_2$ for some $g_1, g_2 \in \mathsf{N_{cF}}$ or
2. $U_1 = Rg_1$ and $U_2 = g_2$ for some $R \in \mathsf{N_{aF}} \cup \mathsf{N_{rR}}$ and $g_1, g_2 \in \mathsf{N_{cF}}$.

It is not hard to see (c.f. [14]) that every $\mathbb{Q}$-$\mathcal{SHIQ}$-concept can be converted into an equivalent one in NNF. In what follows, we use $\sim C$ to denote the result of converting $\neg C$ to NNF. The following lemma shows that we can even assume $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts to be in PNF.

Lemma 4. *Satisfiability of $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts can be polynomially reduced to satisfiability of $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts in PNF.*

Proof We first define an auxiliary mapping and then use this mapping to translate $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts into equivalent ones in PNF. Let C be a $\mathbb{Q}$-$\mathcal{SHIQ}$-concept. For every feature path $u = f_1 \cdots f_n g$ used in C, we assume that $[g], [f_n g], \ldots, [f_1 \cdots f_n g]$ are concrete features not used in C. We inductively define a mapping λ from concrete paths u in C to concepts as follows:

$$\begin{aligned} \lambda(g) &= \top \\ \lambda(fu) &= (\exists [fu], f[u].=) \sqcap \exists f.\lambda(u) \end{aligned}$$

For every $\mathbb{Q}$-$\mathcal{SHIQ}$-concept C, a corresponding concept $\rho(C)$ is obtained by

- first replacing all subconcepts $\forall u_1, u_2.P$ where $u_i = f_1^{(i)} \cdots f_{k_i}^{(i)} g_i$ for $i \in \{1,2\}$ with

$$\forall f_1^{(1)}.\cdots\forall f_{k_1}^{(1)}.\forall g_1, g_1.\neq \sqcup \forall f_1^{(2)}.\cdots\forall f_{k_2}^{(2)}.\forall g_2, g_2.\neq \sqcup \exists u_1, u_2.P$$

- and then replacing all subconcepts $\exists u_1, u_2.P$ with $\exists[u_1], [u_2].P \sqcap \lambda(u_1) \sqcap \lambda(u_2)$.

Now let C be a $\mathbb{Q}$-$\mathcal{SHIQ}$-concept. Using the rewriting rules from [14], we can convert C into an equivalent concept C' in NNF. It is then easy to check that C' is satisfiable iff $\rho(C')$ is satisfiable. Moreover, $\rho(C')$ is clearly in PNF and the translation can be done in polynomial time. ❑

Intuitively, Lemma 4 states that Variant (i) of the binary concrete domain constructors discussed in the previous section can be reduced to the forms $\exists f g_1, g_2.P$ and $\exists g_1, g_2.P$. Variant (ii) of the binary concrete domain constructors does not need to be manipulated in order to fit into the PNF scheme. Let us remark that our algorithm's need for PNF is the reason why we cannot handle arbitrary paths inside the binary concrete domain constructors: it is an interesting exercise to check that the constructor $\forall U_1, U_2.P$ with $U_1 = R_1 \cdots R_n g$ and $U_2 = S_1 \cdots S_m g'.P$ can be reduced to the forms $\exists R g_1, g_2.P$ and $\exists g_1, g_2.P$ if $P \in \{<, \leq, =, \geq, >\}$ but *not* if P is "$\neq$".

3.2 Automata and Constraint Graphs

At the core of the decision procedure to be developed are so-called looping tree-automata, i.e., finite automata on infinite trees for which every run is accepting [23; 21].

Definition 5. Let M be a set and $k \geq 1$. A *k-ary M-tree* is a mapping $T : \{1, \ldots, k\}^* \to M$ that labels each node $\alpha \in \{1, \ldots, k\}^*$ with $T(\alpha) \in M$. Intuitively, the node αi is the i-th child of α. We use ϵ to denote the empty word (corresponding to the root of the tree).

A *looping automaton* $\mathcal{A} = (Q, M, I, \Delta)$ for k-ary M-trees is defined by a finite set Q of *states*, a finite alphabet M, a subset $I \subseteq Q$ of *initial states*, and a *transition relation* $\Delta \subseteq Q \times M \times Q^k$.

A *run* of $\mathcal{A}$ on an M-tree T is a mapping $r : \{1, \ldots, k\}^* \to Q$ with $r(\epsilon) \in I$ and

$$(r(\alpha), T(\alpha), r(\alpha 1), \ldots, r(\alpha k)) \in \Delta$$

for each $\alpha \in \{1, \ldots, k\}^*$. The language $L(\mathcal{A})$ of M-trees *accepted* by $\mathcal{A}$ is

$$L(\mathcal{A}) := \{T \mid \text{there is a run of } \mathcal{A} \text{ on } T\}.$$

Vardi and Wolper [23] show that the emptiness problem for looping automata, i.e., the problem to decide whether $L(\mathcal{A}) = \emptyset$ for a given looping automaton $\mathcal{A}$, is decidable in polynomial time.

We now introduce constraint graphs. As already noted, such graphs will be used to represent the "numerical part" of $\mathbb{Q}$-$\mathcal{SHIQ}$-interpretations in Hintikka-trees.

Definition 6. A *constraint graph* is a directed graph $G = (V, E, \tau)$, where V is a countable set of *nodes*,

$$E \subseteq V \times V \times \{<, \leq, =, \neq, \geq, >\}$$

is a set of *labeled edges*, and

$$\tau \subseteq V \times \{P_q \mid P \in \{<, \leq, =, \neq, \geq, >\} \text{ and } q \in \mathbb{Q}\}$$

is a node labeling relation. In what follows, we sometimes write $\tau(v)$ for $\{P_q \mid (v, P_q) \in \tau\}$.

A constraint graph $G = (V, E, \tau)$ is called *satisfiable over S*—where S is a set equipped with a total ordering $<$—iff there exists a total mapping δ from V to S such that

1. $\delta(v)\, P\, q$ for all $P_q \in \tau(v)$ and
2. $\delta(v_1)\, P\, \delta(v_2)$ for all $(v_1, v_2, P) \in E$.

Such a mapping δ is called a *solution* for G.

We will see later that every Hintikka-tree T induces a constraint graph which represents the "numerical part" of the canonical interpretation described by T. As should be intuitively clear, these induced constraint graphs have to be satisfiable in order for Hintikka-trees to be proper abstractions of interpretations. Since, later on, we must define looping automata which accept exactly the Hintikka-trees for a concept C and role hierarchy $\mathcal{R}$, such automata should be able to verify the satisfiability of (induced) constraint graphs. This check is the main problem to be solved when developing an automata-based decision procedure for $\mathbb{Q}$-$\mathcal{SHIQ}$-concept satisfiability: the induced constraint graph and its satisfiability are "global" notions while automata work "locally". This problem can be overcome as follows: first, we define Hintikka trees such that their induced constraint graphs have a certain form (we will call such constraint graphs *normal*); second, we formulate an adequate criterion for the satisfiability of normal constraint graphs; and third, we show how this criterion can be verified by "local tests" that can be performed by automata. Let us start with introducing normal constraint graphs and the criterion for their satisfiability, which is called *consistency*.

Definition 7. Let $G = (V, E, \tau)$ be a constraint graph. G is called *normal* if it satisfies the following conditions:

1. $(v_1, v_2, P) \in E$ implies $P \in \{<, =\}$,

2. $(v, P_q) \in \tau$ implies $P \in \{<, =, >\}$,
3. for each rational number q appearing in τ and each node $v \in V$, we have $(v, P_q) \in \tau$ for some $P \in \{<, =, >\}$.

Let $\oplus_n$ denote addition modulo n. A $<$-cycle O in a normal constraint graph G is a a finite non-empty sequence of nodes $v_0, \ldots, v_{k-1} \in V$ such that (i) for all $i < k$, there exists a P such that $(v_i, v_{i \oplus_k 1}, P) \in E$ and (ii) there exists an $i < k$ such that $(v_i, v_{i \oplus_k 1}, <) \in E$.

A normal constraint graph G is *consistent* iff it satisfies the following conditions:

1. G contains no $<$-cycle,
2. for all $v \in V$, there exists a $q \in \mathbb{Q}$ such that qPq' for all $P_{q'} \in \tau(v)$,
3. for all $(v_1, v_2, P) \in V$, there exist $q_1, q_2 \in \mathbb{Q}$ such that
   - $q_1 \, P \, q_2$,
   - $q_1 \, P' \, q$ for all $P'_q \in \tau(v_1)$, and
   - $q_2 \, P' \, q$ for all $P'_q \in \tau(v_2)$.

It may appear that Property 3 of consistency is too weak since it only demands the existence of rationals q_1, q_2 for *each* edge between v_1 and v_2 separately instead of for *all* such edges simultaneously: a normal constraint graph with set of edges $\{(v_1, v_2, <), (v_1, v_2, =)\}$ is clearly unsatisfiable, but does not violate Property 3. This, however, is compensated by Property 1 which is violated in this example.

One can show that consistency is indeed an adequate criterion for the satisfiability of normal constraint graphs.

Theorem 8. *A normal constraint graph G is satisfiable over $\mathbb{Q}$ iff G is consistent.*

It is interesting to note that Theorem 8 also holds if satisfiability over $\mathbb{R}$ is considered instead of satisfiability over $\mathbb{Q}$ (the same proof works). However, as noted in [17], Theorem 8 does *not* hold if satisfiability over non-dense structures such as $\mathbb{N}$ is considered.

Intuitively, every constraint graph $G = (V, E, \tau)$ can be converted into a normal one (called a *normalization* of G) by first specializing the relations in E and τ such that Conditions 1 and 2 of normality are satisfied and then augmenting τ such that Condition 3 holds.

Definition 9 (Normalization). A constraint graph $G = (V, E, \tau)$ is a *normalization* of the constraint graph $G' = (V, E', \tau')$ iff it is normal and the following conditions are satisfied:

1. $(v_1, v_2, P) \in E'$ with $P \in \{<, =\}$ implies $(v_1, v_2, P) \in E$,
2. $(v_1, v_2, >) \in E'$ implies $(v_2, v_1, <) \in E$,
3. $(v_1, v_2, \leq) \in E'$ implies $\{(v_1, v_2, <), (v_1, v_2, =)\} \cap E \neq \emptyset$,
4. $(v_1, v_2, \geq) \in E'$ implies $\{(v_2, v_1, <), (v_1, v_2, =)\} \cap E \neq \emptyset$,
5. $(v_1, v_2, \neq) \in E'$ implies $\{(v_1, v_2, <), (v_2, v_1, <)\} \cap E \neq \emptyset$,
6. if $(v, P_q) \in \tau$, then there exists a $v' \in V$ and a P' such that $(v', P'_q) \in \tau'$,
7. $(v, P_q) \in \tau'$ with $P \in \{<, =, >\}$ implies $(v, P_q) \in \tau$,
8. $(v, \leq_q) \in \tau'$ implies $\{(v, <_q), (v, =_q)\} \cap \tau \neq \emptyset$,
9. $(v, \geq_q) \in \tau'$ implies $\{(v, >_q), (v, =_q)\} \cap \tau \neq \emptyset$, and
10. $(v, \neq_q) \in \tau'$ implies $\{(v, <_q), (v, >_q)\} \cap \tau \neq \emptyset$.

Due to Theorem 8 and the obvious fact that a constraint graph is satisfiable iff one of its normalizations is satisfiable, a constraint graph G is satisfiable iff it has a consistent normalization. This property will play an important role for establishing the correspondence between Hintikka-trees and canonical interpretations.

4 Defining Hintikka-trees

In this section, we define Hintikka-trees, which are, as has already been noted, abstractions of canonical (tree-shaped) interpretations. Let us start with defining, for each concept C (in PNF) and role hierarchy $\mathcal{R}$, the set of concepts $\mathsf{cl}(C, \mathcal{R})$ that are "relevant" for deciding whether a given interpretation is a model of C and $\mathcal{R}$: for a given concept C and role hierarchy $\mathcal{R}$, we use $\mathsf{cl}(C, \mathcal{R})$ to denote the smallest set such that

1. $C \in \mathsf{cl}(C, \mathcal{R})$,
2. $\top \in \mathsf{cl}(C, \mathcal{R})$,
3. if $\forall R.D \in \mathsf{cl}(C, \mathcal{R})$, $\mathsf{Trans}(S)$, and $S \sqsubseteq R$, then $\forall S.D \in \mathsf{cl}(C, \mathcal{R})$, and
4. $\mathsf{cl}(C, \mathcal{R})$ is closed under subformulas and $\sim$ (c.f. the remark below Definition 3).

Note that $\sharp\mathsf{cl}(C, \mathcal{R})$ is linear in the length of C and the number of role inclusions in $\mathcal{R}$.

Hintikka-trees are defined in several steps. We start with introducing Hintikka-sets, which form the basis for the definition of so-called Hintikka-labels. As the name indicates, Hintikka-labels are used as node labels in Hintikka-trees. We then define Hintikka-tuples,

which are tuples of Hintikka-labels that describe a valid label configuration for a node and its direct successors in a Hintikka-tree (Hintikka-tuples will also be rather convenient for defining looping automata that accept Hintikka-trees). Eventually, we use Hintikka-labels and Hintikka-tuples to define Hintikka-trees.

Intuitively, each node α of a Hintikka-tree T describes a domain element x of the corresponding canonical model $\mathcal{I}$. The node label of α consists of several parts, one of them a Hintikka-set. This Hintikka-set contains all concepts D from $\mathsf{cl}(C,\mathcal{R})$ such that $x \in D^{\mathcal{I}}$.

Definition 10 (Hintikka-set). Let C be a concept in PNF and $\mathcal{R}$ a role hierarchy. A set $\Psi \subseteq \mathsf{cl}(C,\mathcal{R})$ is a *Hintikka-set for C and $\mathcal{R}$* iff it satisfies the following conditions:

(S1) if $C_1 \sqcap C_2 \in \Psi$, then $\{C_1, C_2\} \subseteq \Psi$,

(S2) if $C_1 \sqcup C_2 \in \Psi$, then $\{C_1, C_2\} \cap \Psi \neq \emptyset$,

(S3) $\{A, \neg A\} \not\subseteq \Psi$ for all concept names A,

(S4) if $f \in \mathsf{N_{aF}}$ is used in C or $\mathcal{R}$, then $(\leqslant 1\ f\ \top) \in \Psi$,

(S5) if $(\leqslant n\ R\ D) \in \mathsf{cl}(C,\mathcal{R})$, then $\{D, \sim D\} \cap \Psi \neq \emptyset$,

(S6) $\top \in \Psi$

Concepts of the form $\exists R.D$, $(\geqslant n\ R\ D)$, $(\leqslant n\ R\ D)$, and $\exists R g_1, g_2.P$ may appear either *marked* or *unmarked* in Ψ.

The marking of concepts is a technical trick that allows us to deal with the inverse role constructor. Intuitively, edges of a Hintikka-tree T describe role successor relationships of the corresponding canonical model $\mathcal{I}$. If $\exists R.D$ occurs in the Hintikka-set of a node β, then there has to exist a "witness" for this concept: either (i) there exists a successor γ of β such that the edge from β to γ represents an R role relationship and D is in the Hintikka-set of γ, or (ii) β has a predecessor α, the edge from α to β represents an $\mathsf{Inv}(R)$ role relationship, and D occurs in the Hintikka-set of α. The marking of concepts is used for bookkeeping of these two possibilities: if $\exists R.D$ occurs *marked* in the Hintikka-set of β, then its predecessor is a "witness" for $\exists R.D$ and we do not need to enforce the existence of a witness among β's successors. The marking of $(\geqslant n\ R\ D)$, $(\leqslant n\ R\ D)$, and $\exists R g_1, g_2.P$ concepts can be explained similarly. Hintikka-sets are one of the components of Hintikka-labels:

Definition 11 (Hintikka-label). Let C be a concept in PNF and $\mathcal{R}$ a role hierarchy. A *Hintikka-label $(\Psi, \omega, V, E, \tau)$ for C and $\mathcal{R}$* consists of

1. a Hintikka-set Ψ for C and $\mathcal{R}$,
2. a set $\omega \subseteq \mathsf{ROL}$ of roles occurring in C or $\mathcal{R}$, and
3. a constraint graph (V, E, τ) where $V \subseteq \mathsf{N_{cF}}$, every $g \in V$ occurs in C, and every q appearing in τ occurs in C.

such that

(N1) if $\exists g_1, g_2.P \in \Psi$, then $g_1, g_2 \in V$ and $(g_1, g_2, P) \in E$,

(N2) if $\forall g_1, g_2.P \in \Psi$ and $g_1, g_2 \in V$, then $(g_1, g_2, P) \in E$,

(N3) if $\exists g.P_q \in \Psi$, then $g \in V$ and $(g, P_q) \in \tau$,

(N4) $R \in \omega$ and $R \sqsubseteq S$ implies $S \in \omega$,

(N5) if $g_1, g_2 \in V$, then $\{(g_1, g_2, <), (g_1, g_2, =), (g_2, g_1, <)\} \cap E \neq \emptyset$,

(N6) if q appears in C, then, for each $g' \in V$, there exists a $P' \in \{<, =, >\}$ such that $(g', P'_q) \in \tau$.

The set of all Hintikka-labels for C and $\mathcal{R}$ is denoted by $\Gamma_{C,\mathcal{R}}$.

Let us explain the intuition behind Hintikka-labels. If α is a node in a Hintikka-tree T, $\mathcal{I}$ the canonical model corresponding to T, and $x \in \Delta_{\mathcal{I}}$ the domain element associated with α, then the Hintikka-label $L = (\Psi, \omega, V, E, \tau)$ of α is a description of x in $\mathcal{I}$. More precisely, (i) the Hintikka-set Ψ is the set of concepts $D \in \mathsf{cl}(C,\mathcal{R})$ such that $x \in D^{\mathcal{I}}$ (ii) ω is the set of roles $R \in \mathsf{ROL}$ such that $(y, x) \in R^{\mathcal{I}}$, where y is the domain element corresponding to the precessor β of α in T; and (iii) the constraint graph (V, E, τ) describes the numerical successors of x and their relationships: if, for some $g \in \mathsf{N_{cF}}$, we have $g \in V$, then $g^{\mathcal{I}}(x)$ is defined. By **(N5)** and **(N6)**, (V, E, τ) fixes the relationship between any two numerical successors of x as well as the relationship between any numerical successor of x and any rational number q appearing in the input concept. By **(N1)**, **(N2)**, and **(N3)**, the relationships stated by E and τ are "consistent" with the Hintikka-set Ψ.

It is rather important that the constraint graph (V, E, τ) fixes the relationship between *any* two nodes of V: as already noted in Section 3, every Hintikka-tree T induces a (normal) constraint graph $G(T)$ that describes the "numerical part" of the canonical interpretation corresponding to T, and should thus be satisfiable. Since $G(T)$ is normal, by Theorem 8 it suffices to demand that $G(T)$ should be consistent. The *complete* determination of the relationships between nodes of the constraint graphs (V, E, τ) in Hintikka-labels will allow us to ensure the consistency of $G(T)$

using a *local* condition which can be verified by looping automata. This condition is part of the definition of Hintikka-tuples, which are introduced next.

Definition 12 (Tuple-graph, Hintikka-tuple). Let C be a concept in PNF and $\mathcal{R}$ a role hierarchy. With $\mathsf{b}_{C,\mathcal{R}}$, we denote

$$\sharp\{D \in \mathsf{cl}(C,\mathcal{R}) \mid D = \exists R.E \text{ or } D = \exists R g_1, g_2.P\} + \sum_{(\geqslant n\ R\ C) \in \mathsf{cl}(C,\mathcal{R})} n.$$

Let $\chi = (L_0, \ldots, L_{\mathsf{b}_{C,\mathcal{R}}})$ be an $\mathsf{b}_{C,\mathcal{R}}$ + 1-tuple of Hintikka-labels with $L_i = (\Psi_i, \omega_i, V_i, E_i, \tau_i)$ for $i \leq \mathsf{b}_{C,\mathcal{R}}$. A constraint graph $G = (V, E, \tau)$ is a *tuple-graph* for χ if

$$\begin{aligned} V &= V_0 \cup \{ig \mid 1 \leq i \leq \mathsf{b}_{C,\mathcal{R}} \text{ and } g \in V_i\} \\ E &\supseteq E_0 \cup \{(ig_1, ig_2, P) \mid 1 \leq i \leq \mathsf{b}_{C,\mathcal{R}} \text{ and } (g_1, g_2, P) \in E_i\} \\ \tau &= \tau_0 \cup \{(ig, P_q) \mid 1 \leq i \leq \mathsf{b}_{C,\mathcal{R}} \text{ and } (g, P_q) \in \tau_i\} \end{aligned}$$

such that

(G1) if $\exists R g, g'.P$ is unmarked in Ψ_0, then there exists an i with $1 \leq i \leq \mathsf{b}_{C,\mathcal{R}}$ such that $ig, g' \in V$, $R \in \omega_i$, and $(ig, g', P) \in E$,

(G2) if $\exists R g, g'.P$ is marked in Ψ_i with $1 \leq i \leq \mathsf{b}_{C,\mathcal{R}}$, then $g, ig' \in V$, $\mathsf{Inv}(R) \in \omega_i$, and $(g, ig', P) \in E$,

(G3) if $\forall R g, g'.P \in \Psi_0$, $R \in \omega_i$, $g \in V_i$, and $g' \in V_0$ for some i with $1 \leq i \leq \mathsf{b}_{C,\mathcal{R}}$, then $(ig, g', P) \in E$,

(G4) if $\forall R g, g'.P \in \Psi_i$, $\mathsf{Inv}(R) \in \omega_i$, $g \in V_0$, and $g' \in V_i$ for some i with $1 \leq i \leq \mathsf{b}_{C,\mathcal{R}}$, then $(g, ig', P) \in E$.

The tuple χ is a *Hintikka-tuple* iff the following conditions are satisfied:

(M1) if $\exists R.D$ is unmarked in Ψ_0, then there exists an i with $1 \leq i \leq \mathsf{b}_{C,\mathcal{R}}$ such that $R \in \omega_i$ and $D \in \Psi_i$,

(M2) if $(\geqslant n\ R\ D) \in \Psi_0$, then either

- $(\geqslant n\ R\ D)$ is unmarked in Ψ_0 and there exists a set $I \subseteq \{1, \ldots, \mathsf{b}_{C,\mathcal{R}}\}$ of cardinality n such that, for each $i \in I$, we have $R \in \omega_i$ and $D \in \Psi_i$ or
- $(\geqslant n\ R\ D)$ is marked in Ψ_0 and there exists a set $I \subseteq \{1, \ldots, \mathsf{b}_{C,\mathcal{R}}\}$ of cardinality $n-1$ such that, for each $i \in I$, we have $R \in \omega_i$ and $D \in \Psi_i$,

(M3) if $\exists R.D$ or $(\geqslant n\ R\ D)$ is marked in Ψ_i with $1 \leq i \leq \mathsf{b}_{C,\mathcal{R}}$, then $\mathsf{Inv}(R) \in \omega_i$ and $D \in \Psi_0$,

(M4) if $\forall R.D \in \Psi_0$ and $R \in \omega_i$ with $1 \leq i \leq \mathsf{b}_{C,\mathcal{R}}$, then $D \in \Psi_i$,

(M5) if $\forall R.D \in \Psi_i$ and $\mathsf{Inv}(R) \in \omega_i$ with $1 \leq i \leq \mathsf{b}_{C,\mathcal{R}}$, then $D \in \Psi_0$,

(M6) if $\forall R.D \in \Psi_0$, $S \in \omega_i$ with $1 \leq i \leq \mathsf{b}_{C,\mathcal{R}}$, $\mathsf{Trans}(S)$, and $S \sqsubseteq R$, then $\forall S.D \in \Psi_i$,

(M7) if $\forall R.D \in \Psi_i$ and $\mathsf{Inv}(S) \in \omega_i$ with $1 \leq i \leq \mathsf{b}_{C,\mathcal{R}}$, $\mathsf{Trans}(S)$, and $S \sqsubseteq R$, then $\forall S.D \in \Psi_0$,

(M8) if $(\leqslant n\ R\ D) \in \Psi_0$, then either

- $(\leqslant n\ R\ D)$ is unmarked in Ψ_0 and the cardinality of the set $\{i \mid 1 \leq i \leq \mathsf{b}_{C,\mathcal{R}}, R \in \omega_i \text{ and } D \in \Psi_i\}$ is at most n or
- $(\leqslant n\ R\ D)$ is marked in Ψ_0 and the cardinality of the set $\{i \mid 1 \leq i \leq \mathsf{b}_{C,\mathcal{R}}, R \in \omega_i \text{ and } D \in \Psi_i\}$ is at most $n-1$,

(M9) if $D \in \Psi_0$, $\mathsf{Inv}(R) \in \omega_i$, and $(\leqslant n\ R\ D) \in \Psi_i$ for $1 \leq i \leq \mathsf{b}_{C,\mathcal{R}}$, then $(\leqslant n\ R\ D)$ is marked in Ψ_i,

(M10) there exists a tuple-graph for χ that has a consistent normalization.

Except for **(M10)**, which refers to tuple-graphs and is the aforementioned local condition enforcing consistency of induced constraint graphs, the properties of Hintikka-tuples should be quite easy to understand. Before we discuss tuple graphs and **(M10)** in more detail, let us introduce Hintikka-trees.

Definition 13 (Hintikka-tree). An $\mathsf{b}_{C,\mathcal{R}}$-ary $\Gamma_{C,\mathcal{R}}$-tree T with $T(\epsilon) = (\Psi_\epsilon, \omega_\epsilon, V_\epsilon, E_\epsilon, \tau_\epsilon)$ is a *Hintikka-tree for* C and $\mathcal{R}$ iff it satisfies the following conditions:

(T1) $C \in \Psi_\epsilon$,

(T2) all concepts in Ψ_ϵ are unmarked, and

(T3) for all $\alpha \in \{1, \ldots, \mathsf{b}_{C,\mathcal{R}}\}^*$, the tuple $(T(\alpha), T(\alpha 1), \ldots, T(\alpha \mathsf{b}_{C,\mathcal{R}}))$ is a Hintikka-tuple.

Let T be a Hintikka-tree, $\alpha \in \{1, \ldots, \mathsf{b}_{C,\mathcal{R}}\}^*$ a node in T, and $T(\alpha) = (\Psi, \omega, V, E, \tau)$. We use $\Psi_T(\alpha)$ to denote Ψ and ω_T to denote ω.

We can now return to the discussion of Property **(M10)**. As is apparent from their definition, tuple-graphs are built by taking the union of all the constraint graphs that appear as a part of the Hintikka-labels in a Hintikka-tuple. The constraint graph $G(T)$ induced by a Hintikka-tree T, in turn, is constructed from tuple-graphs: by **(T3)**, for each node α of T, the tuple

$$\chi_T(\alpha) := (T(\alpha), T(\alpha 1), \ldots, T(\alpha \mathsf{b}_{C,\mathcal{R}}))$$

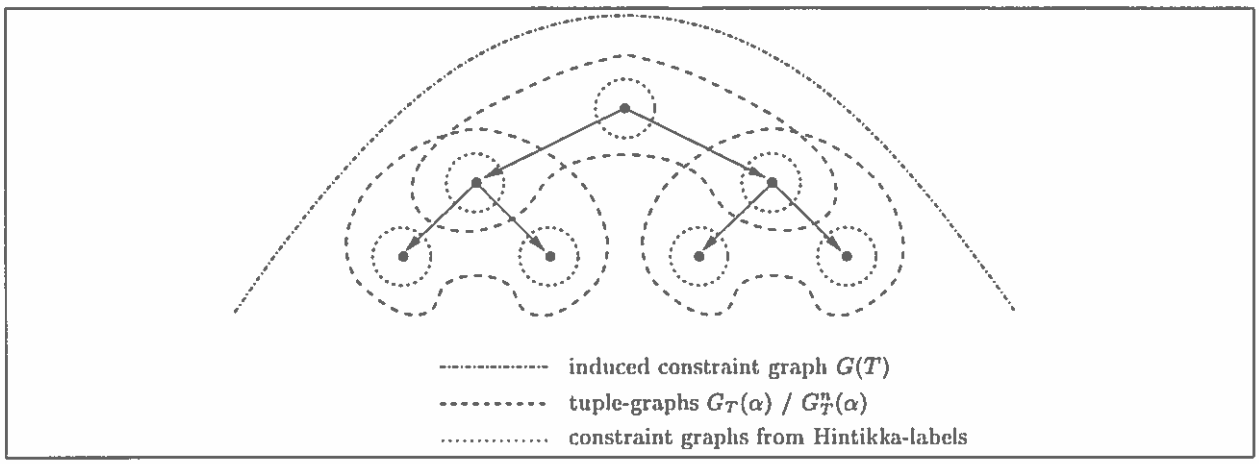


Figure 2: Hintikka-trees and constraint graphs.

is a Hintikka-tuple. By **(M10)**, there exists a tuple-graph $G_T(\alpha)$ for $\chi_T(\alpha)$ which has a consistent normalization $G^n_T(\alpha)$. Modulo some technical details, the constraint graph $G(T)$ induced by T can be viewed as the union of the constraint graphs $G^n_T(\alpha)$ for all nodes α of T. Figure 2 illustrates the relationship between the various constraint graphs involved. In [14], we prove that the consistency of the normalizations $G^n_T(\alpha)$ enforced by **(M10)** implies consistency of the constraint graph $G(T)$ (which is necessary for T to be a proper abstraction of a $\mathbb{Q}$-$\mathcal{SHIQ}$-interpretation). The hardest part of this proof is to show that $G(T)$ satisfies Property 1 of consistency, i.e., that it contains no $<$-cycle: for this proof, it is crucial that

1. the tuple-graph $G_T(\alpha)$ overlaps with the tuple-graph $G_T(\beta)$ if β is a successor of α in T, and
2. the constraint graphs (V, E, τ), which are part of Hintikka-tuples and thus used to build of tuple-graphs, fix the relationship between *any* two elements of V as discussed above.

Using the fact that the constraint graph induced by Hintikka-trees is consistent, the following, central lemma can be established:

Lemma 14. *A concept C in PNF and a role hierarchy $\mathcal{R}$ have a model iff they have a Hintikka-tree.*

5 Defining Looping Automata

To prove decidability of $\mathbb{Q}$-$\mathcal{SHIQ}$-concept satisfiability, it remains to define a looping automaton $\mathcal{A}_{C,\mathcal{R}}$ for each concept C and role hierarchy $\mathcal{R}$ such that $\mathcal{A}_{C,\mathcal{R}}$ accepts exactly the Hintikka-trees for C and $\mathcal{R}$. Using the notion of Hintikka-tuples from Definition 12, this is rather straightforward.

Definition 15. Let C be a concept in PNF, $\mathcal{R}$ a role hierarchy, and $\mathsf{b}_{C,\mathcal{R}}$ as in Definition 12. The looping automaton $\mathcal{A}_{C,\mathcal{R}} = (Q, M, \Delta, I)$ is defined as follows:

- $Q := M := \Gamma_{C,\mathcal{R}}$
- $I := \{(\Psi, \omega, V, E, \tau) \in Q \mid C \in \Psi$ and all concepts in Ψ are unmarked $\}$.
- $\Delta \subseteq Q^{\mathsf{b}_{C,\mathcal{R}}+2}$ such that $(L, L', L_1, \ldots, L_{\mathsf{b}_{C,\mathcal{R}}}) \in \Delta$ iff
  - $L = L'$ and
  - $(L, L_1, \ldots, L_{\mathsf{b}_{C,\mathcal{R}}})$ is a Hintikka-tuple.

As a consequence of the following lemma and Lemma 14, we can reduce satisfiability of concepts (in PNF) w.r.t. role hierarchies to the emptiness of the language accepted by looping automata.

Lemma 16. *T is a Hintikka-tree for C and $\mathcal{R}$ iff $T \in L(\mathcal{A}_{C,\mathcal{R}})$.*

It is an immediate consequence of Lemmas 4, 14, and 16 and the decidability of the emptiness problem of looping automata [23] that satisfiability of $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts w.r.t. role hierarchies is decidable. However, the presented automata-based algorithm additionally provides us with a tight complexity bound if the numbers inside number restrictions are assumed to be encoded unarily: an ExpTime upper bound is obtained by showing that the size of $\mathcal{A}_{C,\mathcal{R}}$ is at most exponential in the size of C and $\mathcal{R}$ and noting that the emptiness problem for looping automata is in PTime [23]. The ExpTime lower bound is an immediate consequence of the fact that $\mathcal{SHIQ}$-concept satisfiability is already ExpTime-hard [22].

Theorem 17. *If numbers inside number restrictions are encoded unarily, then satisfiability of $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts w.r.t. role hierarchies is* EXPTIME*-complete.*

Since subsumption can be reduced to (un)satisfiability, $\mathbb{Q}$-$\mathcal{SHIQ}$-concept subsumption w.r.t. role hierarchies is also EXPTIME-complete.

6 Future Work

In this paper, we have presented the Description Logic $\mathbb{Q}$-$\mathcal{SHIQ}$, which extends the well-known DL $\mathcal{SHIQ}$ with several concrete domain concept constructors that allow the adequate representation of numerical knowledge. As argued in the introduction, $\mathbb{Q}$-$\mathcal{SHIQ}$ is a contribution to several interesting application areas. However, we regard the work presented in this paper only as a first step. As discussed in [14], there exist many possible future research problems connected with the $\mathbb{Q}$-$\mathcal{SHIQ}$ Description Logic. Let us highlight three of them:

(1) To make $\mathbb{Q}$-$\mathcal{SHIQ}$ available for use in applications, modern DL systems like FaCT and RACER, which are implementations of the $\mathcal{SHIQ}$ Description Logic, need to be extended to $\mathbb{Q}$-$\mathcal{SHIQ}$. Unfortunately, the results presented in this paper cannot immediately be used for this task: the aforementioned DL systems are based on tableau-style algorithms while the decision procedure described in this paper is automata-based. Hence, it would be interesting to devise a tableau-based algorithm for $\mathbb{Q}$-$\mathcal{SHIQ}$-concept satisfiability. As discussed in [15] in the context of $\mathcal{TDL}$, the automata-based algorithm presented in this paper can provide important information (i.e., a "regular model property") for this task.

(2) If $\mathbb{Q}$-$\mathcal{SHIQ}$ is to be used for reasoning about ER diagrams as sketched in the introduction, one is usually not interested in the satisfiability of concepts in arbitrary models, but rather in the satisfiability in finite models [3]. These two problems do not coincide since $\mathcal{SHIQ}$, and hence also $\mathbb{Q}$-$\mathcal{SHIQ}$, lacks the finite model property [11]. Thus, it is worthwhile to investigate the decidability and complexity of finite model reasoning with $\mathbb{Q}$-$\mathcal{SHIQ}$.

(3) For some applications, it is desirable to refer to natural numbers instead of rational numbers. As a simple example, consider the concept

$$\begin{aligned}&\exists(\text{left-neighbor numchild}).=_2\\ &\sqcap\ \exists(\text{left-neighbor numchild}),(\text{numchild}).<\\ &\sqcap\ \exists(\text{right-neighbor numchild}).=_3\\ &\sqcap\ \exists(\text{numchild}),(\text{right-neighbor numchild}).<,\end{aligned}$$

where numchild is a concrete feature representing the number of children. Clearly, the above concept should be unsatisfiable. In $\mathbb{Q}$-$\mathcal{SHIQ}$, however, this concept is satisfiable since, in a model, the number of children of the described person may e.g. be 2.5. It would thus be interesting to add a concept constructor $\exists g.\mathsf{nat}$ to $\mathbb{Q}$-$\mathcal{SHIQ}$ expressing that the filler of the concrete feature g is a natural number. If the extended logic should be decidable at all, then at least it seems to require some serious modifications of the presented decision procedure: as noted in [17] in the context of $\mathcal{TDL}$, Theorem 8 does *not* hold if the satisfiability of constraint graphs over non-dense structures such as $\mathbb{N}$ is considered. However, if $\mathbb{Q}$-$\mathcal{SHIQ}$ is extended with an $\exists g.\mathsf{nat}$ constructor, then concepts of the resulting logic can clearly be used to describe constraint graphs all of whose nodes are labeled with the nat predicates. This means that, effectively, we have to decide satisfiability of these constraint graphs over $\mathbb{N}$.

We should like to note that, in many aspects, $\mathbb{Q}$-$\mathcal{SHIQ}$ is already on the border to undecidability: for example, it seems rather unlikely that any kind of arithmetics can be added to $\mathbb{Q}$-$\mathcal{SHIQ}$ without losing decidability. More precisely, it follows from results in [18; 16] that the addition of a concept constructor expressing the addition of two numbers already yields undecidability of reasoning.

Acknowledgements

I would like to thank Ulrike Sattler for helpful comments.

References

[1] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 452–457, Sydney, Australia, 1991.

[2] F. Baader and P. Hanschke. Extensions of concept languages for a mechanical engineering application. In *Proceedings of the 16th German AI-Conference (GWAI-92)*, volume 671 of *Lecture Notes in Computer Science*, pages 132–143. Springer-Verlag, 1992.

[3] D. Calvanese. *Unrestricted and Finite Model Reasoning in Class-Based Representation Formalisms.* Dottorato di ricerca in informatica, Università degli Studi di Roma "La Sapienza", Italia, 1996.

[4] D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In

J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, pages 229–263. Kluwer Academic Publisher, 1998.

[5] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Journal for Intelligent and Cooperative Information Systems*, 2(4):375–399, 1993.

[6] D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, M. Erdmann, and M. Klein. OIL in a nutshell. In *Proceedings of the European Knowledge Acquisition Conference (EKAW 2000)*, volume 1937 of *Lecture Notes In Artificial Intelligence*. Springer-Verlag, 2000.

[7] V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 161–166. Morgan-Kaufmann, 2001.

[8] P. Hanschke. Specifying role interaction in concept languages. In W. Nebel, Bernhard; Rich, Charles; Swartout, editor, *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 318–329. Morgan Kaufmann, 1992.

[9] I. Horrocks and P. Patel-Schneider. The generation of DAML+OIL. In C. Goble, D. L. McGuinness, R. Möller, and P. F. Patel-Schneider, editors, *Proceedings of the International Workshop in Description Logics 2001 (DL2001)*, number 49 in CEUR-WS (http://ceur-ws.org/), pages 30–35, 2001.

[10] I. Horrocks and P. F. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, 1999.

[11] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3), 1999.

[12] I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}(D)$ description logic. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 199–204. Morgan-Kaufmann, 2001.

[13] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3):239–264, 2000.

[14] C. Lutz. Adding numbers to the $\mathcal{SHIQ}$ description logic—first results. LTCS-Report 01-07, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2000. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[15] C. Lutz. Interval-based temporal reasoning with general TBoxes. LTCS-Report 00-06, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2000. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[16] C. Lutz. *The Complexity of Reasoning with Concrete Domains*. PhD thesis, Teaching and Research Area for Theoretical Computer Science, RWTH Aachen, 2001.

[17] C. Lutz. Interval-based temporal reasoning with general TBoxes. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 89–94. Morgan-Kaufmann, 2001.

[18] C. Lutz. NExpTime-complete description logics with concrete domains. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR'01)*, number 2083 in Lecture Notes in Artifical Intelligence, pages 45–60. Springer-Verlag, 2001.

[19] C. Lutz. Reasoning about entity relationship diagrams with constraints on attributes. Submitted to the 2002 Workshop on Description Logics, 2002.

[20] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[21] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 4, pages 133–191. Elsevier Science Publishers B. V., 1990.

[22] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, 2001.

[23] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logic of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986.

Approximation and Difference in Description Logics*

Sebastian Brandt
Theoretical Computer Science
RWTH Aachen
sbrandt@cs.rwth-aachen.de

Ralf Küsters
Theoretical Computer Science
CAU Kiel
kuesters@ti.informatik.uni-kiel.de

Anni-Yasmin Turhan
Theoretical Computer Science
RWTH Aachen
turhan@cs.rwth-aachen.de

Abstract

Approximation is a new inference service in Description Logics first mentioned by Baader, Küsters, and Molitor. Approximating a concept, defined in one Description Logic, means to translate this concept to another concept, defined in a second typically less expressive Description Logic, such that both concepts are as closely related as possible with respect to subsumption. The present paper provides the first in-depth investigation of this inference task. We prove that approximations from the Description Logic $\mathcal{ALC}$ to $\mathcal{ALE}$ always exist and propose an algorithm computing them.

As a measure for the accuracy of the approximation, we introduce a syntax-oriented difference operator, which yields a concept that contains all aspects of the approximated concept that are not present in the approximation. It is also argued that a purely semantical difference operator, as introduced by Teege, is less suited for this purpose. Finally, for the logics under consideration, we propose an algorithm computing the difference.

1 Introduction

Approximation in Description Logics (DLs) was first mentioned by Baader, Küsters, and Molitor [2] as a possible new inference problem. The present paper is the first to investigate this problem in depth. Informally, approximation is defined as follows: given a concept C defined in a DL $\mathcal{L}_s$ ("s" for source) find a concept D, the upper/lower approximation of C, in a DL $\mathcal{L}_d$ ("d" for destination) such that i) D subsumes/is subsumed by C, and ii) D is a minimal/maximal concept in $\mathcal{L}_d$ (w.r.t. subsumption) with this property. Throughout this paper we will mainly focus on upper approximations. There are a number of different applications of this inference problem, some of which we will mention here (see [7] for others, such as knowledge-base vivification, an application already mentioned in [5, 9]).

Translation of knowledge-bases. Approximation can be used to (automatically) translate a knowledge-base written in an expressive DL into another (semantically closely related) knowledge-base in a less expressive DL. The translation may become necessary to port knowledge-bases between different knowledge representation systems or to integrate different knowledge-bases.

Non-standard inferences for expressive DLs. Non-standard inferences in DLs, such as computing the least common subsumer (lcs), matching and unification of concepts, have been introduced to support the construction and maintenance of DL knowledge-bases (see [6] for an overview). However, up to now they are mostly restricted to quite inexpressive DLs, for example to those that do not allow for concept disjunction. Approximation can be used to overcome this problem to some extent. For example, the existing matching algorithms can be lifted up to handle more expressive DLs as follows: instead of directly matching concept patterns (defined in a small DL) against concepts (defined in a DL that can not be handled by existing matching algorithms), one can first approximate the concept (in the small DL) and then match against its approximation. Even though some information may be lost, e.g., the matcher is more general than the correct one, the accuracy of the result may still suffice.

* This work has been partially supported by the Deutsche Forschungsgemeinschaft, DFG Project BA 1122/4-1.

Another example, which was in fact the main motivation for us to investigate approximation in the first place, is the *computation of commonalities between concepts*. This inference service is used in our chemical process engineering application [14] to support the bottom-up construction of knowledge-bases [6, 1]. Typically, the lcs is employed to accomplish this task. Formally, the lcs of two concepts, say C_1 and C_2, defined in some DL $\mathcal{L}$, is the most specific concept (w.r.t. subsumption) in $\mathcal{L}$ that subsumes both concepts. In case $\mathcal{L}$ allows for concept disjunction, the lcs is just the disjunction of C_1 and C_2 ($C_1 \sqcup C_2$). Thus, the problem is that a user inspecting this concept does not learn anything about the actual commonalities between C_1 and C_2. By using approximation, however, one can make the commonalities explicit by first approximating C_1 and C_2 in a sublanguage of $\mathcal{L}$ which does not allow to express concept disjunction, and then computing the lcs of the approximations in this sublanguage.

Supporting frame-based user interfaces of DL systems. In the interaction with DL systems, users with little knowledge representation expertise may have difficulties to understand and make use of the full expressive power of the underlying DLs. To overcome this problem, some knowledge representation systems have been equipped with a simplified frame-based user interface built on top of a more powerful DL system. Examples for such systems are the TAMBIS system [3] and the ontology editor OilEd [4] built on top of the FaCT DL system [11]. On many occasions, these systems have to present concept descriptions to the user for editing, inspection, or as a solution of inference problems. Such concept descriptions, however, need not always fit into the restricted representation of the frame-based user interface or may overwhelm an inexperienced user. In such cases, approximation can be helpful as a means to represent concept descriptions in a simplified fashion suited to the user interface and the users' level of expertise.

The main technical result of this paper (Section 3) is to show that concept descriptions defined in the standard DL $\mathcal{ALC}$, which allows for concept conjunction and disjunction, value and existential restrictions, and full negation, can be approximated (from above) in the DL $\mathcal{ALE}$, a DL that does not allow for concept disjunction and full negation.

Once one has given an (upper) approximation D of C a natural question regards the loss of information, i.e., what aspects of C are not captured by D. Therefore we propose a difference operator, which given C (in $\mathcal{L}_s$) and D (in $\mathcal{L}_d$) yields a concept E (the difference of C and D) in $\mathcal{L}_s$ such that E conjoint with D is equivalent to C ($E \sqcap D \equiv C$). In other words, E contains the information that is missing in the approximation D of C. Such an operator has already been defined by Teege [16]. He requires that E is the most general concept description in $\mathcal{L}_s$ w.r.t. subsumption that satisfies the above equivalence. However, as we will see, such a purely semantical definition of difference yields very unintuitive concepts. We therefore propose a new syntax-based definition, which better captures the intuition behind difference. Roughly speaking, the difference E between C and D will be obtained by syntactically removing those parts of C that are already present in D. In Section 4, we provide a formal definition and give a heuristic algorithm for computing the difference between an $\mathcal{ALC}$- and an $\mathcal{ALE}$-concept description.

In Section 5 we present some experiences with our prototypical implementations of the algorithms presented here and conclude with some remarks on future work. Full proofs of the main theoretical results are provided in the appendix. All proofs and further details can be found in our technical report [7].

2 Description Logics

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 DL $\mathcal{ALE}$, concept descriptions are formed using the constructors top-concept ($\top$), concept conjunction ($C \sqcap D$), existential restriction ($\exists r.C$), value restriction ($\forall r.C$), primitive negation ($\neg A$), and the bottom-concept ($\bot$). The DL $\mathcal{ALC}$ additionally provides us with concept disjunction ($C \sqcup D$) and full negation ($\neg C$). Note that in $\mathcal{ALC}$ every concept description can be negated whereas in $\mathcal{ALE}$ negation is only allowed in front of concept names. For a DL $\mathcal{L}$, such as $\mathcal{ALE}$ and $\mathcal{ALC}$, a concept description formed with the constructors allowed in $\mathcal{L}$ is called *$\mathcal{L}$-concept description* in the following.

As usual, the semantics of a concept description 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 $A \in N_C$ to a set $A^I \subseteq \Delta$ 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 *sub-*

Syntax	Semantics	$\mathcal{ALE}$	$\mathcal{ALC}$
$\top$	Δ	x	x
$C \sqcap D$	$C^I \cap D^I$	x	x
$\exists r.C$	$\{x \in \Delta \mid \exists y : (x,y) \in r^I \wedge y \in C^I\}$	x	x
$\forall r.C$	$\{x \in \Delta \mid \forall y : (x,y) \in r^I \rightarrow y \in C^I\}$	x	x
$\neg A$, $A \in N_C$	$\Delta \setminus A^I$	x	x
$\bot$	$\emptyset$	x	x
$C \sqcup D$	$C^I \cup D^I$		x
$\neg C$	$\Delta \setminus C^I$		x

Table 1: Syntax and semantics of concept descriptions.

sumed by the description D ($C \sqsubseteq D$) iff $C^I \subseteq D^I$ holds for all interpretations $\mathcal{I}$; C and D are *equivalent* ($C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$; C is strictly subsumed by D ($C \sqsubset D$) iff $C \sqsubseteq D$ and $C \not\equiv D$. Subsumption and equivalence are PSPACE-complete in $\mathcal{ALC}$ [15] and NP-complete in $\mathcal{ALE}$ [10].

In order to approximate $\mathcal{ALC}$-concept descriptions by $\mathcal{ALE}$-concept descriptions, we will need to compute the least common subsumer in $\mathcal{ALE}$.

Definition 1 *Given $\mathcal{L}$-concept descriptions $C_1, \ldots, C_n$, for some description logic $\mathcal{L}$, the $\mathcal{L}$-concept description C is the* least common subsumer *(lcs) of $C_1, \ldots, C_n$ ($C = \mathsf{lcs}(C_1, \ldots, C_n)$ for short) iff (i) $C_i \sqsubseteq C$ for all $1 \leq i \leq n$, and (ii) C is the least concept description with this property, i.e., if C' satisfies $C_i \sqsubseteq C'$ for all $1 \leq i \leq n$, then $C \sqsubseteq C'$.*

Depending on the DL under consideration, the lcs of two or more concept descriptions need not always exist, but if it exists, then, by definition, it is unique up to equivalence. For instance, in $\mathcal{ALC}$ the lcs trivially exists since $lcs(C, D) \equiv C \sqcup D$. For $\mathcal{ALE}$, which does not allow for concept disjunction, the existence is not obvious. However, as shown in [1], the lcs of two or more $\mathcal{ALE}$-concept descriptions always exists, its size may grow exponentially in the size of the input descriptions, and it can be computed in exponential time.

3 Computing Approximations

In this section, we show how $\mathcal{ALC}$-concept descriptions can be approximated (from above) by $\mathcal{ALE}$-concept descriptions. Let us first define the notion of approximation formally.

Definition 2 *Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two DLs, and let C be an $\mathcal{L}_1$- and D be an $\mathcal{L}_2$-concept description. Then, D is called an* upper (lower) $\mathcal{L}_2$-approximation *of C ($D = \mathsf{approx}_{\mathcal{L}_2}(C)$ for short) if (i) $C \sqsubseteq D$ ($D \sqsubseteq C$), and (ii) D is minimal (maximal) with this property, i.e., $C \sqsubseteq D'$ ($D' \sqsubseteq C$) and $D' \sqsubseteq D$ ($D \sqsubseteq D'$) implies $D' \equiv D$ for all $\mathcal{L}_2$-concept descriptions D'.*

Note that approximations need not exist in general. Consider for example the DLs $\mathcal{L}_1 = \{\sqcap\}$ and $\mathcal{L}_2 = \{\sqcup\}$, i.e., the DLs that have no $\bot$ or $\top$ and only allow for concept conjunction and concept disjunction, respectively. Let A and B denote concept names. Then, there does not exist an upper $\mathcal{L}_1$-approximation of the $\mathcal{L}_2$-concept description $A \sqcup B$. Conversely, there does not exist a lower $\mathcal{L}_2$-approximation of the $\mathcal{L}_1$-concept description $A \sqcap B$. Also note that approximations need not be uniquely determined. For example, both A and B are lower $\mathcal{L}_1$-approximations of $A \sqcup B$ with $\mathcal{L}_1$ defined as above.

In this paper, we restrict our investigations to upper approximations. Therefore, whenever we speak of approximations in the following, we mean upper approximations. Moreover, we concentrate on upper $\mathcal{ALE}$-approximations of $\mathcal{ALC}$-concept descriptions. Since $\mathcal{ALE}$ allows for concept conjunction it immediately follows that if upper $\mathcal{ALE}$-approximations exist, they are uniquely determined up to equivalence: If D_1 and D_2 are two upper $\mathcal{ALE}$-approximations of the same $\mathcal{ALC}$-concept, then so is $D_1 \sqcap D_2$. But then, by definition of upper approximation, $D_1 \sqcap D_2 \sqsubseteq D_1$ and $D_1 \sqcap D_2 \sqsubseteq D_2$ implies $D_1 \sqcap D_2 \equiv D_1 \equiv D_2$.

3.1 The Naïve Approximation Approach

Now, let us turn to the question of how upper $\mathcal{ALE}$-approximations can be computed from $\mathcal{ALC}$-concept descriptions. We first present a naïve approach to this problem and show that it fails. This will then motivate the definition of the (correct) approximation algorithm.

It is easy to see that, given an $\mathcal{ALC}$-concept description $C = E \sqcup F$ with $\mathcal{ALE}$-concept descriptions E and F, the $\mathcal{ALE}$-approximation of C is $\mathsf{lcs}(E, F)$. Having observed

this, one might think that every $\mathcal{ALC}$-concept description C can be approximated by simply replacing every concept disjunction in C by the lcs operator and evaluating the lcs operators from inside out. However, the $\mathcal{ALC}$-concept description

$$C_{\text{ex},1} = (\forall r.B \sqcup (\exists r.B \sqcap \forall r.A)) \sqcap \exists r.A,$$

with concept names A and B, illustrates that this is not the case: The obtained approximation would be $\mathsf{lcs}(\forall r.B, (\exists r.B \sqcap \forall r.A)) \sqcap \exists r.A \equiv \top \sqcap \exists r.A \equiv \exists r.A$. However, as one can easily check, $C_{\text{ex},1} \sqsubseteq \exists r.(A \sqcap B) \sqsubset \exists r.A$. In fact, $\exists r.(A \sqcap B)$ is the correct upper $\mathcal{ALE}$-approximation of $C_{\text{ex},1}$.

As it turns out, we will have to turn the concept descriptions into a certain normal form before substituting disjunctions by the lcs. Roughly speaking, the normal forms are obtained by distributing concept conjunctions over concept disjunctions. In the example, this yields the concept description $(\forall r.B \sqcap \exists r.A) \sqcup (\exists r.B \sqcap \forall r.A \sqcap \exists r.A)$ and replacing the disjunction by the lcs yields $\mathsf{lcs}(\forall r.B \sqcap \exists r.A, \exists r.B \sqcap \forall r.A \sqcap \exists r.A) \equiv \exists r.(A \sqcap B)$, which is the correct result.

Still, the following example illustrates that normalizing concepts in this way does not suffice in the general case. The description

$$C_{\text{ex},2} = \exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B)$$

is already in normal form, but substituting the concept disjunction with the lcs yields $\exists r.A \sqcap \exists r.B \sqcap \forall r.\mathsf{lcs}(\neg A, \neg B) \equiv \exists r.A \sqcap \exists r.B \sqcap \forall r.\top \equiv \exists r.A \sqcap \exists r.B$. However, the $\mathcal{ALE}$-approximation of $C_{\text{ex},2}$ is $\exists r.(A \sqcap \neg B) \sqcap \exists r.(B \sqcap \neg A)$. The reason is that we need to propagate value restrictions to existential restrictions in order to obtain the correct approximations of the existential restrictions.

In what follows, we will first introduce the normal forms and then present the approximation algorithm, which works on these normal forms and does the propagation on-the-fly.

3.2 $\mathcal{ALC}$-Normal form

For the sake of simplicity, we assume that the set N_R of role names is the singleton $\{r\}$. However, all definitions and results can easily be generalized to arbitrary sets of role names. We also assume that each conjunction in an $\mathcal{ALE}$-concept description contains at most one value restriction of the form $\forall r.C'$ (this is w.l.o.g. due to the equivalence $\forall r.E \sqcap \forall r.F \equiv \forall r.(E \sqcap F)$). Some notation is needed to access the different parts of an $\mathcal{ALE}$-concept description C (and an $\mathcal{ALC}$-concept description where disjunction only occurs within value or existential restrictions): $\mathsf{prim}(C)$ denotes the set of all (negated) concept names and the bottom concept occurring on the top-level conjunction of C; if there exists a value restriction of the form $\forall r.C'$ on the top-level conjunction of C, then $\mathsf{val}(C) := C'$; otherwise, $\mathsf{val}(C) := \top$; $\mathsf{ex}(C) := \{C' \mid \text{there exists } \exists r.C' \text{ on the top-level conjunction of } C\}$.

Definition 3 *An $\mathcal{ALC}$-concept description C is in $\mathcal{ALC}$-*normal form *iff*

1. *if $C \equiv \bot$, then $C = \bot$; if $C \equiv \top$, then $C = \top$;*
2. *otherwise, C is of the form $C = C_1 \sqcup \cdots \sqcup C_n$ with*

$$C_i = \bigsqcap_{A \in \mathsf{prim}(C_i)} A \sqcap \bigsqcap_{C' \in \mathsf{ex}(C_i)} \exists r.C' \sqcap \forall r.\mathsf{val}(C_i),$$

$C_i \not\equiv \bot$, and $\mathsf{val}(C_i)$ and every concept description in $\mathsf{ex}(C_i)$ is in $\mathcal{ALC}$-normal form, for all $i = 1, \ldots, n$.

Obviously, every $\mathcal{ALC}$-concept description can be turned into an equivalent concept description in $\mathcal{ALC}$-normal form. Unfortunately, this may take exponential time, as the example $(A_1 \sqcup A_2) \sqcap \cdots \sqcap (A_{2n-1} \sqcup A_{2n})$ shows, whose $\mathcal{ALC}$-normal form is of size exponential in n.

3.3 The Approximation Algorithm

Our approximation algorithm is based on the following structural characterization of subsumption between an $\mathcal{ALC}$-concept description C in $\mathcal{ALC}$-normal form and an $\mathcal{ALE}$-concept description D. The idea is that D is compared to every disjunct C_i in C. This comparison in turn is very similar to the structural characterization of subsumption between $\mathcal{ALE}$-concept descriptions [1]. For the full proof, see [7].

Theorem 4 *Let C be an $\mathcal{ALC}$-concept description in $\mathcal{ALC}$-normal form (as specified in Definition 3) and D an $\mathcal{ALE}$-concept description. Then, $C \sqsubseteq D$ iff $C \equiv \bot$, or $D \equiv \top$, or for all $i = 1, \ldots, n$:*

1. *$\mathsf{prim}(D) \subseteq \mathsf{prim}(C_i)$, and*
2. *for all $D' \in \mathsf{ex}(D)$ there exists $C' \in \mathsf{ex}(C_i)$ such that $C' \sqcap \mathsf{val}(C_i) \sqsubseteq D'$, and*
3. *$\mathsf{val}(C_i) \sqsubseteq \mathsf{val}(D)$.*

The approximation algorithm, denoted by c-approx$_{\mathcal{ALE}}$, is depicted in Figure 1. Given C, it finds an $\mathcal{ALE}$-concept description which is as specific as possible and satisfies the conditions of

Input: $\mathcal{ALC}$-concept description C
Output: upper $\mathcal{ALE}$-approximation of C

1. If $C \equiv \bot$, then $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) := \bot$; if $C \equiv \top$, then $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) := \top$
2. Otherwise, transform C into $\mathcal{ALC}$-normal form $C_1 \sqcup \cdots \sqcup C_n$ and return $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) :=$

$$\prod_{A \in \bigcap_{i=1}^{n} \mathsf{prim}(C_i)} A \sqcap \prod_{(C_1', \ldots, C_n') \in \mathsf{ex}(C_1) \times \cdots \times \mathsf{ex}(C_n)} \exists r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i' \sqcap \mathsf{val}(C_i)) \mid 1 \le i \le n\} \sqcap \forall r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val}(C_i)) \mid 1 \le i \le n\}$$

Figure 1: The recursive algorithm $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C)$.

Theorem 4. For $C \equiv \bot$ and $C \equiv \top$ this is trivial. In case $C \not\equiv \bot$ and $D := \mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) \not\equiv \top$, one needs to show that i) the Conditions 1, 2, and 3 of Theorem 4 are satisfied for C and D, and that ii) D is a minimal concept description with this property. (See Appendix A for the full proof).

Theorem 5 *For every $\mathcal{ALC}$-concept description C the $\mathcal{ALE}$-approximation exists, is uniquely determined up to equivalence, and can be computed by* $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$, *i.e.,* $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) \equiv \mathsf{approx}_{\mathcal{ALE}}(C)$.

In order to apply the algorithm $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ to our examples, we first transform $C_{\mathsf{ex},1}$ into $\mathcal{ALC}$-normal form:

$$C_{\mathsf{ex},1} \equiv \big((\exists r.A \sqcap \forall r.B) \sqcup (\exists r.A \sqcap \exists r.B \sqcap \forall r.A)\big).$$

Next, we compute the actual approximation of this concept description:

$$\begin{aligned}
&\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_{\mathsf{ex},1}) \\
&\equiv \exists r.\mathsf{lcs}\big(\mathsf{c\text{-}approx}_{\mathcal{ALE}}(A \sqcap B), \mathsf{c\text{-}approx}_{\mathcal{ALE}}(A \sqcap A)\big) \sqcap \\
&\quad \exists r.\mathsf{lcs}\big(\mathsf{c\text{-}approx}_{\mathcal{ALE}}(A \sqcap B), \mathsf{c\text{-}approx}_{\mathcal{ALE}}(A \sqcap B)\big) \sqcap \\
&\quad \forall r.\mathsf{lcs}\big(\mathsf{c\text{-}approx}_{\mathcal{ALE}}(B), \mathsf{c\text{-}approx}_{\mathcal{ALE}}(A)\big) \\
&\equiv \exists r.\mathsf{lcs}(A \sqcap B, A) \sqcap \exists r.\mathsf{lcs}(A \sqcap B, A \sqcap B) \sqcap \\
&\quad \forall r.\mathsf{lcs}(B, A) \\
&\equiv \exists r.A \sqcap \exists r.(A \sqcap B) \sqcap \forall r.\top \\
&\equiv \exists r.(A \sqcap B)
\end{aligned}$$

As one can see, the existential restriction is now correctly approximated by the algorithm $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$. Our second example $C_{\mathsf{ex},2}$ is already in $\mathcal{ALC}$-normal form, therefore its approximation is directly obtained by the following steps:

$$\begin{aligned}
&\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_{\mathsf{ex},2}) \\
&\equiv \exists r.\mathsf{c\text{-}approx}_{\mathcal{ALE}}\big(A \sqcap (\neg A \sqcup \neg B)\big) \sqcap \\
&\quad \exists r.\mathsf{c\text{-}approx}_{\mathcal{ALE}}\big(B \sqcap (\neg A \sqcup \neg B)\big) \sqcap \\
&\quad \forall r.\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\neg A \sqcup \neg B) \\
&\equiv \exists r.\mathsf{c\text{-}approx}_{\mathcal{ALE}}(A \sqcap \neg B) \sqcap \\
&\quad \exists r.\mathsf{c\text{-}approx}_{\mathcal{ALE}}(B \sqcap \neg A) \sqcap \forall r.\top \\
&\equiv \exists r.(A \sqcap \neg B) \sqcap \exists r.(B \sqcap \neg A)
\end{aligned}$$

One can see that unlike the naïve approach the $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ algorithm propagates the value restrictions and therefore yields the correct approximation.

In [1], it has been shown that the lcs of two $\mathcal{ALE}$-concept descriptions can grow exponentially in the size of the given concept descriptions. Since $\mathsf{approx}_{\mathcal{ALE}}(E \sqcup F) \equiv \mathsf{lcs}(E, F)$ for $\mathcal{ALE}$-concept descriptions E and F, it immediately follows that the $\mathcal{ALE}$-approximation can grow exponentially as well. Moreover, one can show that $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ runs in double exponential time (see Appendix B for details).

Corollary 6 *The $\mathcal{ALE}$-approximation of $\mathcal{ALC}$-concept descriptions may grow exponentially and there is a double-exponential time algorithm computing it.*

Whether or not there also exists an exponential time approximation algorithm is an open problem.

4 The Difference Operator

In the previous section we have seen how to compute the $\mathcal{ALE}$-approximation of a given $\mathcal{ALC}$-concept description. For such a pair C, D of approximated and approximating concept, a very natural question regards the loss of information, i.e., what aspects of C are not captured by D.

An answer to this question requires a notion of the "difference" between concept descriptions. For instance, a comparison between the example concept $C_{ex,2}$ and its approximation $\exists r.(A \sqcap \neg B) \sqcap \exists r.(B \sqcap \neg A)$ should reveal that the value restriction $\forall r.(\neg A \sqcup \neg B)$ is not captured by the approximation.

A first definition of a difference operator was proposed by Teege [16] who declares the difference $C - D$ of two given $\mathcal{L}$-concept descriptions with $C \sqsubseteq D$ as

$$max\{E \in \mathcal{L} \mid E \sqcap D \equiv C\},$$

where the maximum is defined with respect to subsumption. Since $\mathcal{ALC}$ provides full negation, a most general concept E with $E \sqcap D \equiv C$ is always $C \sqcup \neg D$. For the difference between the example concept $C_{ex,2}$ and its approximation Teege's difference operator would consequently return

$$(\exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B)) \\ \sqcup \neg(\exists r.(A \sqcap \neg B) \sqcap \exists r.(B \sqcap \neg A)),$$

which obviously does not help a human user to ascertain the information lost by the approximation.

The example suggests to look for a syntactic minimum instead of a semantic maximum in order to find a compact representation of the difference of two concepts. Such a syntactic minimum will be defined by the so-called subdescription ordering.

4.1 Subdescription Ordering

In [12, 2], a *subdescription ordering* on $\mathcal{ALE}$-concept descriptions has been proposed to deal with syntactical redundancies. In order to extend this to our case we need to introduce an analogous ordering on $\mathcal{ALC}$-concepts. The idea is to obtain a subdescription of some $\mathcal{ALC}$-concept description C by means of two kinds of modifications. Firstly, by making inconsistencies explicit; and secondly, by removing disjuncts and conjuncts, and by replacing some existential or value restrictions by their respective subdescriptions. Formally, this leads to the following definition.

Definition 7 *Let C, D be $\mathcal{ALC}$-concept descriptions in $\mathcal{ALC}$-normal form. Let $C = C_1 \sqcup \cdots \sqcup C_n$. Then, D is a* subdescription of C *($D \preceq_d C$) iff $D = \bot$ or D is obtained from C by performing some of the following steps:*

1. *Remove some disjuncts C_i for $1 \leq i \leq n$,*
2. *for every remaining C_i:*
   (a) *remove some conjuncts $A \in \mathsf{prim}(C_i)$,*
   (b) *remove some conjuncts $\exists r.C_i'$ with $C_i' \in \mathsf{ex}(C_i)$,*
   (c) *remove the conjunct $\forall r.\mathsf{val}(C_i)$,*
   (d) *for every remaining $C_i' \in \mathsf{ex}(C_i) \cup \{\mathsf{val}(C_i)\}$: replace C_i' by C_i'' with $C_i'' \preceq_d C_i'$*

If everything is removed from C, the resulting concept is $\top$. As an example, consider the equivalent concept descriptions $C := \exists r.A \sqcap \forall r.\neg B$ and $D := (\exists r.(A \sqcup B) \sqcap \forall r.\neg B) \sqcup (\exists r.\neg A \sqcap \forall r.A)$. By removing the last disjunct from D and the last disjunct in the remaining existential restriction we find $C \preceq_d D$.

Based on the subdescription ordering, we can provide the new definition of the difference operator.

Definition 8 *Let C be an $\mathcal{ALC}$-concept description and D an $\mathcal{ALE}$-concept description. The* difference *$C - D$ of C and D is defined as a minimal (w.r.t. $\preceq_d$) $\mathcal{ALC}$-concept description E with $E \sqcap D \equiv C \sqcap D$.*

Intuitively, the idea is to remove all subdescriptions from C which are either redundant within C or already present in D. It should be noted that in case of $C \sqsubseteq D$, and thus, $C \sqcap D \equiv C$, the only difference to Teege's difference operator is that the minimum w.r.t. $\preceq_d$ is used instead of the maximum w.r.t. $\sqsubseteq$. Finally, note that the difference between C and D is not a priori uniquely determined. By abuse of language and notation, we will still refer to *the* difference $C - D$ (see also Theorem 9). Coming back to the example at the beginning of the section, the difference (according to Definition 8) between $C_{ex,2}$ and its approximation is $\forall r.(\neg A \sqcup \neg B)$, as desired.

4.2 The Difference Algorithm

Having defined our difference operator, we need to devise an algorithm to actually compute the difference $C - D$. In [12], an algorithm has been proposed to compute the difference $C - D$ of $\mathcal{ALE}$-concept descriptions C and D. We want to extend this to the case where C is an $\mathcal{ALC}$-concept description and D an $\mathcal{ALE}$-concept description. However, as illustrated by the following example, we must also consider the case where D is an $\mathcal{ALC}$-concept description, because in the difference $C - D$ all redundancies originaly present in C are also removed.

$$C_{ex,3} = \exists r.((A \sqcap C) \sqcup (B \sqcap C)) \sqcap \forall r.(A \sqcup B)$$
$$D_{ex,3} = \top$$

Although $C_{ex,3}$ has "nothing in common" with $D_{ex,3}$, the difference $C_{ex,3} - D_{ex,3}$ does not yield $C_{ex,3}$, but

the smaller (w.r.t. $\preceq_d$) concept $\exists r.C \sqcap \forall r.(A \sqcup B)$ ($\equiv C_{ex,3}$). To obtain this result, the difference of the existential and the value restrictions of $C_{ex,3}$ must be computed, i.e., $E - (A \sqcup B)$ with $E = (A \sqcap C) \sqcup (B \sqcap C)$, which, intuitively, should be C. Hence, computing the difference of two $\mathcal{ALC}$-concepts is necessary for the computation of $C - D$ even if D is only an $\mathcal{ALE}$-concept description.

Handling the Difference of Disjunctions

If a disjunction occurs on the left-hand side of the difference, i.e., C is of the form $C_1 \sqcup \cdots \sqcup C_n$, then the difference $C - D$ can be computed by distributing the difference over the disjunction, i.e., by computing $(C_1 - D) \sqcup \cdots \sqcup (C_n - D)$ first and removing possibly redundant disjuncts then. In case of a disjunction on the right-hand side, i.e., $D = D_1 \sqcup \cdots \sqcup D_n$, one could dually try to compute $C - (D_1 \sqcup \cdots \sqcup D_n)$ as the conjunction $(C - D_1) \sqcap \cdots \sqcap (C - D_n)$.

For the above example, by distributing the difference over the disjunction in E, we obtain $((A \sqcap C) - (A \sqcup B)) \sqcup ((B \sqcap C) - (A \sqcup B))$. Following the above idea, i.e., turning the difference into an conjunction of differences yields $((A \sqcap C) - A) \sqcap ((A \sqcap C) - B) = C \sqcap (A \sqcap C)$, and thus, $(A \sqcap C)$ after simplification. Analogously, one obtains $(B \sqcap C)$ for the second difference. Hence, altogether nothing has been removed from E and we do not obtain C as desired. Unfortunately, similar straightforward approaches, such as decomposing the difference $C - (D_1 \sqcup \cdots \sqcup D_n)$ into the disjunction $(C - D_1) \sqcup \cdots \sqcup (C - D_n)$, also fail. Currently, the question of how to compute differences with disjunctions on the right-hand side remains an open problem.

A Heuristic Approach

As a first step towards a solution, we propose the heuristic algorithm c-diff as depicted in Figure 2. It computes the difference between an $\mathcal{ALC}$-concept description in $\mathcal{ALC}$-normal form and an $\mathcal{ALE}$-concept-description.

If C is a disjunction of subconcepts C_i then the difference between C and D is obtained by firstly computing the differences between the disjuncts and D and then eliminating the semantically redundant resulting disjuncts. Any disjunction occurring on the right-hand side of a difference during the computation is simply omitted; see the definition of E^* in Figure 2, where the difference between an existential restriction $E \in \mathsf{ex}(C)$ and $\mathsf{val}(C) \sqcap \mathsf{val}(D)$ is required.

Despite this straightforward approach, the following three properties still can be shown for every computation of c-diff(C, D). (The full proof can be found in Appendix C).

Theorem 9 *Let C be an $\mathcal{ALC}$-concept description in $\mathcal{ALC}$-normal form and D be an $\mathcal{ALE}$-concept description. Then,*

1. c-diff$(C, D) \sqcap D \equiv C \sqcap D$.
2. *If C is an $\mathcal{ALE}$-concept description, then $C - D$ is uniquely determined modulo associativity and commutativity of concept conjunction, and $C - D$ and* c-diff(C, D) *coincide.*
3. *Given an oracle for subsumption,* c-diff(C, D) *runs in polynomial time in the size of C and D.*

Thus, in case C is an $\mathcal{ALE}$-concept description, c-diff exactly computes the difference $C - D$ (Theorem 9, 2.). If C is an $\mathcal{ALC}$-concept description, we know that c-diff does not remove too much from C (Theorem 9, 1.). However, c-diff might not have computed the exact difference $C - D$.

Nevertheless, the experiences with our prototypical implementation of c-diff show that the algorithm works quite satisfactorily in practice. To illustrate this, consider the simple example concept $C_{ex,1}$ and its $\mathcal{ALE}$-approximation $\exists r.(A \sqcap B)$. In order to compute the difference c-diff$(C_{ex,1}, \exists r.(A \sqcap B))$, $C_{ex,1}$ first has to be transformed into $\mathcal{ALC}$-normal form, yielding $(\forall r.B \sqcap \exists r.A) \sqcup (\exists r.B \sqcap \forall r.A \sqcap \exists r.A)$. We now have to compute c-diff$(\forall r.B \sqcap \exists r.A,\ \exists r.(A \sqcap B))$ and c-diff$(\exists r.B \sqcap \forall r.A \sqcap \exists r.A,\ \exists r.(A \sqcap B))$. For the first expression, Condition 3(i) in the algorithm c-diff causes $\exists r.A$ to be removed. As no other existential restriction is left, the first expression evaluates to $\forall r.B$. The second expression similarly yields $\forall r.A$. We finally obtain $\forall r.A \sqcup \forall r.B$, which is exactly $C_{ex,1} - \text{c-approx}_{\mathcal{ALE}}(C_{ex,1})$. Analogously, one can verify that c-diff$(C_{ex,2}, \text{c-approx}_{\mathcal{ALE}}(C_{ex,2})) = C_{ex,2} - \text{c-approx}_{\mathcal{ALE}}(C_{ex,2}) = \forall r.(\neg A \sqcup \neg B)$.

An Improved Heuristic

Instead of simply omitting $\mathsf{val}(C)$ in the definition of E^* (see Figure 2) in case $\mathsf{val}(C)$ is not an $\mathcal{ALE}$-concept description, one could as well define E^* to be c-diff$(E, \text{c-approx}_{\mathcal{ALE}}(\mathsf{val}(C) \sqcap \mathsf{val}(D)))$, which in many cases comes closer to $\mathsf{val}(C) \sqcap \mathsf{val}(D)$ than only $\mathsf{val}(D)$. (Note that if $\mathsf{val}(C)$ is an $\mathcal{ALE}$-concept description, $\text{c-approx}_{\mathcal{ALE}}(\mathsf{val}(C) \sqcap \mathsf{val}(D)) \equiv \mathsf{val}(C) \sqcap \mathsf{val}(D)$.)

For example, consider the $\mathcal{ALC}$-concept description $E := \exists r.C \sqcap \forall r.((A \sqcap C) \sqcup (B \sqcap C))$. While the first version of c-diff applied to E and $\top$, i.e., c-diff$(E, \top)$, would yield E, the improved version yields $\exists r.\top \sqcap \forall r.((A \sqcap C) \sqcup (B \sqcap C))$, because $\text{c-approx}_{\mathcal{ALE}}((A \sqcap C) \sqcup$

Input: $\mathcal{ALC}$-concept description $C = C_1 \sqcup \cdots \sqcup C_n$ in $\mathcal{ALC}$-normal form,
$\mathcal{ALE}$-concept description D
Output: c-diff(C, D)

1. If $C \sqcap D \equiv \bot$, then c-diff$(C, D) := \bot$;
2. If $n > 1$, then let c-diff$(C, D) := \bigsqcup_{i=1}^{n}$ c-diff(C_i, D) and iteratively remove c-diff(C_j, D) from the disjunction in case c-diff$(C_j, D) \sqsubseteq \bigsqcup_{i \neq j}$ c-diff(C_i, D);
3. If $n = 1$, then c-diff$(C, D) :=$

$$\bigsqcap_{A \in \mathsf{prim}(C) \setminus \mathsf{prim}(D)} A \sqcap \forall r.\mathsf{c\text{-}diff}(\mathsf{val}(C), \mathsf{val}(D)) \sqcap \bigsqcap_{E \in \mathcal{E}'_r} \exists r.E$$

where the value restriction is omitted in case c-diff$(\mathsf{val}(C), \mathsf{val}(D)) \equiv \top$ and $\mathcal{E}'_r$ is computed as follows:

Let $\mathcal{E}_r = \{C'_1, \ldots, C'_n\} := \mathsf{ex}(C)$.
For $i = 1$ to n do begin
  If (i) there exists $C' \in \mathcal{E}_r \setminus \{C'_i\}$ with $\mathsf{val}(D) \sqcap \mathsf{val}(C) \sqcap C' \sqsubseteq C'_i$, or
    (ii) there exists $D' \in \mathsf{ex}(D)$ with $\mathsf{val}(D) \sqcap \mathsf{val}(C) \sqcap D' \sqsubseteq C'_i$
  then $\mathcal{E}_r := \mathcal{E}_r \setminus \{C'_i\}$
end
$\mathcal{E}'_r := \{E^* \mid E \in \mathcal{E}_r\}$ where $E^* :=$ c-diff$(E, \mathsf{val}(C) \sqcap \mathsf{val}(D))$, if $\mathsf{val}(C)$ is an $\mathcal{ALE}$-concept description, and $E^* :=$ c-diff$(E, \mathsf{val}(D))$ otherwise.

Figure 2: The algorithm c-diff(C, D).

$(B \sqcap C)) = C$ and $C - C = \top$. Still, the improved c-diff fails on $C_{ex,3}$ and $D_{ex,3}$, since c-approx$_{\mathcal{ALE}}(A \sqcup B) = \top$, and thus, the value restriction of $C_{ex,3}$ remains unchanged as before. Moreover, due to the additional complexity caused by the computation of the approximation, the algorithm c-diff would no longer be a polynomial time algorithm (with an oracle for subsumption).

5 Prototypical Implementations

We have evaluated a first prototypical implementation of c-approx$_{\mathcal{ALE}}$ realized in Lisp and using the FaCT system [11] as an underlying subsumption tester. Our implementation of c-approx$_{\mathcal{ALE}}$ utilizes the optimized lcs implementation described in [17]. In contrast to the c-approx$_{\mathcal{ALE}}$ algorithm specified in Figure 1 our implementation reduces the number of lcs calls in advance. For many concept descriptions in $\mathcal{ALC}$-normal form it is likely that disjuncts share the same existential restrictions due to the normalization. These existential restrictions cause unnecessary lcs calls when approximating the existential restrictions. Some of the combinations in the Cartesian product of the existential restrictions yield argument sets for the lcs that are supersets of other combinations. These supersets yield more general and therefore redundant lcs concept descriptions. For example, computing the approximation of the concept description $(\exists r.A \sqcup D) \sqcap (\exists r.B \sqcap \exists r.C)$ starts with computing the equivalent $\mathcal{ALC}$-normal form: $(\exists r.A \sqcap \exists r.B \sqcap \exists r.C) \sqcup (D \sqcap \exists r.B \sqcap \exists r.C)$. During the computation of the actual approximation a naive realization of c-approx$_{\mathcal{ALE}}$ induces the lcs calls: lcs(A, B), lcs(A, C), lcs(B, B), lcs(B, C), lcs(C, B), and lcs(C, C). However, only the trivial combinations lcs(B, B) and lcs(C, C) add existential restrictions to the approximation which are not subsumed by the other combinations. Therefore, in this case, the existential restrictions can be obtained without using the lcs at all. So, in order to obtain the correct approximation in general, it suffices to compute the lcs only of those combinations that do not have a subset among the combinations. This method is employed in our implementation: We compute first the minima (w.r.t. subset) of the set of combinations and then apply the lcs to these minima.

We applied c-approx$_{\mathcal{ALE}}$ to $\mathcal{ALC}$-concepts from a TBox derived from our application in chemical process engineering. This application TBox contains 120 concepts and 40 roles. Surprisingly, for our unfolded input concepts with concept sizes up to 740, it turned out that the approximations were always smaller than

their unfolded input concepts. The approximations had an average concept size of 81 and they had just a third of the size of the unfolded input concepts on the average. Each of the test concept descriptions was approximated within less than 3 seconds runtime. Unfortunately, our implementation ran out of memory computing approximations of some randomly generated $\mathcal{ALC}$-concepts of similar size, but consisting of disjunctions with at least 6 disjuncts. In our chemical process engineering application all disjunctions consist of smaller number of disjuncts and therefore it may be likely that these cases do not appear in practice.

Thus, our prototypical implementation of c-approx$_{\mathcal{ALE}}$ indicates that, despite the high theoretical complexity, the approximation inference might be practicable for cases relevant in applications. Further optimizations are of course necessary. Standard optimization techniques as lazy unfolding are very likely to highly improve the performance for run-times as well as for sizes of returned concepts.

We have also implemented a prototype for the c-diff algorithm as presented in Figure 2. For a first evaluation we applied the c-diff implementation to test concepts derived from our process engineering TBox. More precisely, we applied c-diff to the same $\mathcal{ALC}$-concept descriptions used for the evaluation of c-approx$_{\mathcal{ALE}}$ together with their approximations generated by our c-approx$_{\mathcal{ALE}}$ implementation. For these test cases the c-diff implementation returned concept descriptions with an average size of 170 and a maximum size of 630. Thus, it turned out that in many cases the concept size of the difference between the original concept description and its approximation is about 37% of the unfolded and normalized concept itself.

The runtimes for computing the difference took 2 seconds on the average and each difference was computed within 6.5 seconds. Unlike c-approx$_{\mathcal{ALE}}$ this prototypical implementation behaved also well on randomly generated concept descriptions. But for practical applications of this non-standard inference powerful optimizations are still necessary. Moreover, the output concept descriptions need to be smaller and more compact in order to be readable and comprehensible for a human user.

6 Conclusion and Future Work

We have investigated approximation as a new inference problem for DLs. As a main technical result, a double-exponential time algorithm computing upper approximations of $\mathcal{ALC}$-concepts in $\mathcal{ALE}$ has been devised. We have also introduced a syntax-based difference operator to measure the accuracy of approximations and an efficient heuristic algorithm, which uses subsumption testing as an oracle, to actually compute the difference of concepts.

Our first evaluation of the implementations of c-approx$_{\mathcal{ALE}}$ and c-diff indicates that there is need for further optimization. Even more important, since the concepts returned by both algorithms are quite big and hard to read and comprehend by a human user, it is necessary to rewrite the concepts using the concept definitions from the underlying $\mathcal{ALC}$-TBox to obtain smaller concepts. To this purpose, one needs to extend the existing rewriting approach for $\mathcal{ALE}$ [2] to $\mathcal{ALC}$.

Another direction for future work is of course to extend our results to more expressive DLs. In fact our approach presented here can be extended to the approximation of $\mathcal{ALCN}$-concept descriptions by concept descriptions in $\mathcal{ALEN}$ (or sublanguages thereof) as shown in [8]. It remains as future work to extend lcs and approximation to DLs with qualified number restrictions as well as to adapt the difference operator to description logics with (qualified) number restrictions.

References

[1] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In T. Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 96–101. Morgan Kaufmann, 1999.

[2] F. Baader, R. Küsters, and R. Molitor. Rewriting concepts using terminologies. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *KR2000: Principles of Knowledge Representation and Reasoning*, pages 297–308, San Francisco, 2000. Morgan Kaufmann.

[3] P. G. Baker, A. Brass, S. Bechhofer, C. Goble, N. Paton, and R. Stevens. TAMBIS: Transparent access to multiple bioinformatics information sources. In J. Glasgow, T. Littlejohn, F. Major, R. Lathrop, D. Sankoff, and C. Sensen, editors, *6th Int. Conf. on Intelligent Systems for Molecular Biology*, pages 25–34, Montreal, Canada, 1998. AAAI Press, Menlo Park.

[4] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. Oiled: a reason-able ontology editor for the semantic web. In F. Baader, G. Brewka, and Th. Eiter, editors, *Proceedings of the Joint German/Austrian Conference on AI (KI 2001)*,

volume 2174 of *Lecture Notes in Artificial Intelligence*, pages 396–408, Vienna, Austria, 2001. Springer-Verlag.

[5] A. Borgida and D. W. Etherington. Hierarchical knowledge bases and efficient disjunctive reasoning. In H. J. Levesque, R. J. Brachman and R. Reiter, editors, *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 33–43, Toronto, Canada, May 1989. Morgan Kaufmann.

[6] S. Brandt and A.-Y. Turhan. Using non-standard inferences in description logics — what does it buy me? In *Proceedings of the KI-2001 Workshop on Applications of Description Logics (KIDLWS'01)*, number 44 in CEUR-WS, Vienna, Austria, September 2001. RWTH Aachen. Proceedings online available from http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-44/.

[7] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. LTCS-Report 01-06, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2001. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[8] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximating $\mathcal{ALCN}$-Concept Descriptions. See http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html. To appear.

[9] W. W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumers in description logics. In W. Swartout, editor, *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 754–760, San Jose, CA, July 1992. MIT Press.

[10] F. M. Donini, M. Lenzerini, D. Nardi, B. Hollunder, W. Nutt, and A. Spaccamela. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 53(2–3):309–327, 1992.

[11] I. Horrocks. Using an expressive description logic: FaCT or fiction? In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, *KR'98: Principles of Knowledge Representation and Reasoning*, pages 636–645. Morgan Kaufmann, San Francisco, California, 1998.

[12] R. Küsters. *Non-Standard Inferences in Description Logics*, volume 2100 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001.

[13] R. Küsters and R. Molitor. Computing least common subsumers in $\mathcal{ALEN}$. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 219–224. Morgan Kaufman, 2001.

[14] U. Sattler. *Terminological knowledge representation systems in a process engineering application.* PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 1998.

[15] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[16] G. Teege. Making the difference: A subtraction operation for description logics. In P. Torasso J. Doyle, E. Sandewall, editor, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 540–550, Bonn, FRG, May 1994. Morgan Kaufmann.

[17] A.-Y. Turhan and R. Molitor. Using lazy unfolding for the computation of least common subsumers. In *Proceedings of the International Workshop in Description Logics 2001 (DL2001)*, Stanford, USA, August 2001.

Appendix

A Proof of Theorem 5

Without loss of generality, we may assume C in $\mathcal{ALC}$-normal form since (1) the algorithm c-approx$_{\mathcal{ALE}}$ starts by computing the $\mathcal{ALC}$-normal form of its input and (2) $\top$ and $\bot$ are represented uniquely in $\mathcal{ALC}$-normal form.

1. Show $C \sqsubseteq$ c-approx$_{\mathcal{ALE}}(C)$. To this end, we show by structural induction of C that the conditions for subsumption from Theorem 4 hold.

If $C \in \{\bot, \top\}$ then c-approx$_{\mathcal{ALE}}(C) = C$ which trivially satisfies the subsumption conditions. Otherwise, we may assume as induction hypothesis that the claim holds for the subconcepts of C occurring in existential and value restrictions. For C we therefore find that:

- By definition of c-approx$_{\mathcal{ALE}}$, the set prim(c-approx$_{\mathcal{ALE}}(C)$) of primitive concepts equals $\bigcap_{i=1}^{n}$ prim(C_i) which is always a subset of prim(C_i).
- Show: for lcs$\{$c-approx$_{\mathcal{ALE}}(C_i' \sqcap$ val$(C_i)) \mid 1 \leq i \leq n\}$ and for all i there exists some existential restriction $C' \in$ ex(C_i) such that $C' \sqcap$ val(C_i) is

subsumed by $\mathsf{lcs}(\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i' \sqcap \mathsf{val}(C_i)) \mid 1 \leq i \leq n\}$.

Pick $C' = C_i'$. By induction hypothesis it holds that $C' \sqcap \mathsf{val}(C_i)$ is subsumed by the approximation $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i' \sqcap \mathsf{val}(C_i))$. The definition of the lcs now guarantees that $C' \sqcap \mathsf{val}(C_i)$ is also subsumed by $\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i' \sqcap \mathsf{val}(C_i)) \mid 1 \leq i \leq n\}$.

- Show: $\mathsf{val}(C_i) \sqsubseteq \mathsf{val}(\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C))$ for every i. By induction hypothesis we already know that the value restriction $\mathsf{val}(C_i)$ is subsumed by the approximation $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val}(C_i))$ for every i. Consequently, for the lcs we find that $\mathsf{val}(C_i)$ is subsumed by $\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val}(C_i)) \mid 1 \leq i \leq n\}$ for every i.

2. Show $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) \sqsubseteq D$ for all $\mathcal{ALE}$-concept descriptions D with $C \sqsubseteq D$. Without loss of generality, let D be in $\mathcal{ALE}$-normal form. Proof by induction over the structure of C.

If $C \in \{\bot, \top\}$, then $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) = C \sqsubseteq D$.

Otherwise, we may assume that the claim holds for the subconcepts of C occurring in existential and value restrictions. If $D = \top$, then trivially $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) \sqsubseteq D$. Otherwise, the subsumption $C \sqsubseteq D$ induces the following facts:

- $\mathsf{prim}(D) \subseteq \mathsf{prim}(C_i)$ for every i. As the set $\mathsf{prim}(\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C))$ of primitive concepts is defined as the intersection of every $\mathsf{prim}(C_i)$, this implies $\mathsf{prim}(D) \subseteq \mathsf{prim}(\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C))$.
- For all $D' \in \mathsf{ex}(D)$ and for all i there is an existential restriction $C' \in \mathsf{ex}(C_i)$ with $C' \sqcap \mathsf{val}(C_i) \sqsubseteq D'$. The induction hypothesis now guarantees that $C' \sqcap \mathsf{val}(C_i)$ is subsumed by $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C' \sqcap \mathsf{val}(C_i)) \sqsubseteq D$ for every i. Consequently, for the lcs it holds that $\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C' \sqcap \mathsf{val}(C_i)) \mid 1 \leq i \leq n\} \sqsubseteq D$.
- For all i we have $\mathsf{val}(C_i) \sqsubseteq \mathsf{val}(D)$. By induction hypothesis we know that the value restriction $\mathsf{val}(C_i)$ is subsumed by the approximation $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val}(C_i)) \sqsubseteq \mathsf{val}(D)$. Hence, we similarly find $\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val}(C_i)) \mid 1 \leq i \leq n\} \sqsubseteq \mathsf{val}(D)$. ∎

B Proof of Corollary 6

The algorithm $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ expects its input in $\mathcal{ALE}$-normal form. Nevertheless, instead of transforming C into normal form before applying $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ we may also do the necessary transformation on the fly for every role level currently visited.

Let $|C| = n$. The computation of $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C)$ starts by transforming C into $D := C_1 \sqcup \cdots \sqcup C_m$—such that every C_i has no disjunction on the topmost role level—but does not modify the lower role levels. The concept D can thus have exponentially many ($2^{p(n)}$ for some polynomial p) disjuncts on the topmost level each of which is limited in size by n.

According to the recursive structure of $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ the following expressions must be computed:

1. the conjunction $\bigsqcap_{A \in \bigcap_i \mathsf{prim}(C_i)} A$ of primitive concepts;
2. an existential restriction $\exists r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i' \sqcap \mathsf{val}(C_i)) \mid 1 \leq i \leq m\}$ for every tuple $(C_1', \ldots, C_m')$ with $C_i' \in \mathsf{ex}(C_i)$;
3. one value restriction $\forall r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val}(C_i)) \mid 1 \leq i \leq m\}$.

Obviously, Step 1 can be computed in polynomial time in the size of D and thus in exponential time in n.

As D has exponentially many disjuncts C_i, which in turn have a linear number of existential restrictions C_i', the number of existential restrictions to be computed in Step 2 is at most double exponential in n. For every such existential restriction an lcs of a set of exponential cardinality must be computed. Each element of such a set is of the form $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i' \sqcap \mathsf{val}(C_i))$. Hence, $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ is recursively invoked on a concept description of size bounded by the size of C and with a role depth decreased by one. Thus, the computation tree of $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ (with the lcs's not evaluated for the time being), is of size double exponential in the size of C. We need to show that evaluating the lcs's occurring in the computation tree, does not increase the complexity.

We start to evaluate the lcs's from the bottom to the top of the computation tree for $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C)$. Every lcs operation in the tree has an exponential number of arguments and every argument is of size double exponential in $|C|$. Moreover, one can easily show that every argument is not only in $\mathcal{ALE}$-normal form, but also has the following properties:

- It contains no subexpression of the form $P \sqcap \neg P$ (where $P \in N_C$), $E \sqcap \bot$, $E \sqcap \top$, $\exists r.\bot$, or $\forall r.\top$.
- For every subexpression of the form $\exists r.E \sqcap \forall r.F$ it holds that $E \sqsubseteq F$, i.e., the value restriction is propagated into the existential restriction.

This holds because all concepts returned by c-approx$_{\mathcal{ALE}}$ have these properties. As shown in [1], the size of the lcs can therefore be bounded by the product of the sizes of the arguments. Thus, evaluating the lcs's on the bottom level yields concept descriptions of size at most double exponential. This evaluation process is iterated on every level of the computation tree for c-approx$_{\mathcal{ALE}}(C)$ where lcs's occur. Since the depth of this tree is bounded by $|C|$ (more precisely, by the role depth of C), the whole evaluation can be carried out in double-exponential time. ∎

C Proof of Theorem 9

1. Proof by structural induction of C.

$C \in \mathsf{prim}(C)$: Then the conjunction $\mathsf{c\text{-}diff}(C, D) \sqcap D$ is equivalent to $P_{C \setminus D} \sqcap D$, where $P_{C \setminus D}$ is the conjunction over all primitive concepts in $\mathsf{prim}(C) \setminus \mathsf{prim}(D)$. Let $P_{C \cap D}$ denote the conjunction over all primitive concepts in $\mathsf{prim}(C) \sqcap \mathsf{prim}(D)$. As $D \sqsubseteq P_{C \cap D}$, the term $\mathsf{c\text{-}diff}(C, D) \sqcap D$ is still equivalent to the conjunction $P_{C \setminus D} \sqcap P_{C \cap D} \sqcap D$. This expression, however, is equivalent to $C \sqcap D$.

$C = C_1 \sqcup C_2$: Without loss of generality, we may assume exactly two disjuncts on the top-level of C. By induction we have $C_i - D \sqcap D \equiv C_i \sqcap D$ for all $i \in \{1, 2\}$. From this it follows $C - D \sqcap D \equiv ((C_1 - D) \sqcup (C_2 - D)) \sqcap D \equiv ((C_1 - D) \sqcap D) \sqcup ((C_2 - D) \sqcap D) \equiv (C_1 \sqcap D) \sqcup (C_2 \sqcap D) \equiv (C_1 \sqcup C_2) \sqcap D \equiv C \sqcap D$.

No disjunction on the top-level of C: Show $\mathsf{c\text{-}diff}(C, D) \sqcap D \equiv C \sqcap D$. According to the characterization of subsumption (Theorem 4), three conditions must hold for equivalence:

- The set $\mathsf{prim}(\mathsf{c\text{-}diff}(C, D) \sqcap D)$ of primitive concepts is equal to the intersection $\mathsf{prim}(\mathsf{c\text{-}diff}(C, D)) \cap \mathsf{prim}(D)$ which by definition is the intersection of $(\mathsf{prim}(C) \setminus \mathsf{prim}(D))$ and $\mathsf{prim}(D)$. This is equal to $\mathsf{prim}(C) \cap \mathsf{prim}(D)$, the set of primitive concepts in the conjunction $C \sqcap D$.

- By induction hypothesis, the conjunction $\mathsf{c\text{-}diff}(\mathsf{val}(C), \mathsf{val}(D)) \sqcap \mathsf{val}(D)$ is equivalent to $\mathsf{val}(C) \sqcap \mathsf{val}(D)$. By definition, $\mathsf{val}(C \sqcap D)$ is equivalent to $\mathsf{val}(C) \sqcap \mathsf{val}(D)$, which concludes this case.

- Show ($\sqsubseteq$). Consider an existential restriction $F' \in \mathsf{ex}(C \sqcap D)$. We have to find an existential restriction $E' \in \mathsf{ex}(\mathsf{c\text{-}diff}(C, D) \sqcap D)$ with $E' \sqcap \mathsf{val}(\mathsf{c\text{-}diff}(C, D) \sqcap D) \sqsubseteq F'$. From the previous case we know that $\mathsf{val}(\mathsf{c\text{-}diff}(C, D) \sqcap D)$ is equivalent to $\mathsf{val}(C \sqcap D)$. Since $\mathsf{ex}(C \sqcap D)$ is equal to the union $\mathsf{ex}(C) \cup \mathsf{ex}(D)$ we may distinguish two cases.

  If $F' \in \mathsf{ex}(D)$ then we can choose $E' := F'$, because it also occurs in the set $\mathsf{ex}(\mathsf{c\text{-}diff}(C, D) \sqcap D)$ which is the union of the concept descriptions in $\mathsf{ex}(\mathsf{c\text{-}diff}(C, D)) \cup \mathsf{ex}(D)$. We thus find that the conjunction $E' \sqcap \mathsf{val}(\mathsf{c\text{-}diff}(C, D) \sqcap D)$ is subsumed by F'.

  If $F' \in \mathsf{ex}(C) \setminus \mathsf{ex}(D)$, then Conditions (i) and (ii) in the definition of the algorithm $\mathsf{c\text{-}diff}(C, D)$ guarantee that there exists an existential restriction $\tilde{E}' \in \mathsf{ex}(\mathsf{c\text{-}diff}(C, D))$ with the following properties. If $\mathsf{val}(C)$ is an $\mathcal{ALE}$-concept description then $\tilde{E}'$ is of the form $\mathsf{c\text{-}diff}(E', (\mathsf{val}(D) \sqcap \mathsf{val}(C)))$ and the conjunction $E' \sqcap \mathsf{val}(D) \sqcap \mathsf{val}(C)$ is subsumed by F'. According to the induction hypothesis, $\mathsf{c\text{-}diff}(E', (\mathsf{val}(D) \sqcap \mathsf{val}(C))) \sqcap \mathsf{val}(D) \sqcap \mathsf{val}(C)$ is equivalent to $E' \sqcap \mathsf{val}(D) \sqcap \mathsf{val}(C)$. Consequently, we find that $\tilde{E}' \sqcap \mathsf{val}(C) \sqcap \mathsf{val}(D)$ is subsumed by F'. It is easy to see that $\mathsf{val}(C) \sqcap \mathsf{val}(D)$ is equivalent to $\mathsf{val}(C \sqcap D)$ which again is equivalent to $\mathsf{val}(\mathsf{c\text{-}diff}(C, D) \sqcap D)$ as we know from above. Hence, we have found an existential restriction $\tilde{E}'$ such that the conjunction $\tilde{E}' \sqcap \mathsf{val}(\mathsf{c\text{-}diff}(C, D) \sqcap D)$ is subsumed by F'. If D is no $\mathcal{ALE}$-concept description then $\tilde{E}'$ is of the form $E' \sqcap \mathsf{val}(D)$. This case is analogous to the previous one.

  Show ($\sqsupseteq$). In analogy to the case ($\sqsubseteq$), consider some existential restriction $E' \in \mathsf{ex}(\mathsf{c\text{-}diff}(C, D) \sqcap D)$. We have to find an existential restriction $F' \in \mathsf{ex}(C \sqcap D)$ such that $F' \sqcap \mathsf{val}(C \sqcap D)$ is subsumed by E'. Again, we have two cases to discriminate.

  If $E' \in \mathsf{ex}(D)$, then we can again select $F' := E'$ which also occurs in the set $\mathsf{ex}(C \sqcap D)$ of existential restrictions.

  If $E' \in \mathsf{ex}(\mathsf{c\text{-}diff}(C, D)) \setminus \mathsf{ex}(D)$, then there exists $C'_i \in \mathsf{ex}(C)$ such that $E' = \mathsf{c\text{-}diff}(C'_i, F)$ with $F = \mathsf{val}(C) \sqcap \mathsf{val}(D)$ or $F = \mathsf{val}(D)$ (depending on whether $\mathsf{val}(C)$ is an $\mathcal{ALE}$-concept description or not). We choose $F' := C'_i$. By induction, we know $F' \sqcap F \equiv E' \sqcap F$, and thus, $F' \sqcap \mathsf{val}(C \sqcap D) \sqsubseteq E' \sqcap F \sqsubseteq E'$.

2. This result was already shown in [12].

3. The proof can be found in our technical report [7]. ∎

Connecting abstract description systems

Oliver Kutz and Frank Wolter
Institut für Informatik
Universität Leipzig
Augustus-Platz 10–11
04109 Leipzig, Germany
{kutz, wolter}@informatik.uni-leipzig.de

Michael Zakharyaschev
Department of Computer Science
King's College London
Strand
London WC2R 2LS, U.K.
mz@dcs.kcl.ac.uk

Abstract

Combining knowledge representation and reasoning formalisms like description logics (DLs), temporal logics, and logics of space, is worthwhile but difficult. It is worthwhile because usually realistic application domains comprise various aspects of the world, thus requiring suitable combinations of formalisms modeling each of these aspects. It is difficult because the computational behavior of the resulting hybrids is often much worth than the behavior of its components. In this paper we propose a combination method which is robust in the computational sense and still allows for certain interactions between the combined systems. The combination method, called *E-connection*, will be defined and investigated for so-called abstract description systems (ADS) which include all standard description logics, various logics of time and space, modal logics, and epistemic logics. The main theoretical result is that every E-connection of any finite number of decidable ADSs is decidable as well. Four instances of E-connections of ADSs will be discussed: (1) the E-connection of DLs with the logic $\mathcal{MS}$ intended for quantitative reasoning about space, (2) the E-connection of DLs with the logic $\mathsf{S4}_\mathsf{u}$ (containing RCC-8) that can be used for qualitative reasoning about space, (3) the E-connection of two DLs ($\mathcal{ALCO}$ and $\mathcal{SHIQ}$), and (4) the E-connection of DLs with propositional temporal logic PTL and $\mathsf{S4}_\mathsf{u}$.

1 Introduction

Combining knowledge representation and reasoning formalisms is worthwhile but difficult. It is worthwhile because usually realistic application domains comprise various aspects of the world around us, thus requiring suitable combinations of formalisms modeling each of these aspects. It is difficult because the computational behavior of the resulting hybrids is often much worse than the behavior of the combined components; see, e.g., [11, 14, 16]. To be more specific, consider three examples related to description logics.

(i) Classical description logics (DLs) represent knowledge at a rather abstract logical level. To cope with applications which require predefined predicates or temporal and spatial dimensions, combinations of DLs with *concrete domains* such as the natural numbers equipped with predicates like $<$, Allen's interval logic [1], or region connection calculus RCC-8 [29] have been proposed [7, 19]. The addition of a concrete domain to a DL is a rather sensitive operation as far as the preservation of its nice computational properties is concerned: even 'weak' DLs combined with 'weak' concrete domains can become undecidable; see, e.g., [8, 18, 28]. In fact, to investigate DLs with concrete domains is rather hard and requires developing new techniques, cf. [27].

(ii) Standard DLs have been designed to represent *static* knowledge which is time- and agent-independent. To take into account the dynamic aspects of knowledge, DLs have been extended with temporal, dynamic, epistemic and other intentional operators [25, 9, 10, 13, 2, 39, 41]. The resulting formalisms become 'many-dimensional' and sometimes show rather nasty computational behavior: combinations of simple description logics (say, $\mathcal{ALC}$) with simple temporal logics (say, propositional temporal logic PTL) can be highly undecidable [10, 41, 40, 16]. These logics also require new approaches [41, 16], and it is

still unclear whether practical reasoning systems can be developed for many-dimensional logics, cf. [26].

(iii) Often there is a need to combine two or more description logics: while one part of the application domain may require constructors of DL_1, another part can only be represented using constructors of DL_2. Putting the constructors of DL_1 and DL_2 together to form a new DL may result in an undecidable logic, even if both components are decidable. As an example, consider the DLs $\mathcal{ALCF}$ (extending $\mathcal{ALC}$ with functional roles (or features) and the same-as constructor (or agreement) on chains of functional roles) and $\mathcal{ALC}^{+,o,\sqcup}$ (extending $\mathcal{ALC}$ with the transitive closure, composition, and union of roles). For both DLs, the subsumption of concept descriptions is known to be decidable [21, 32, 6]. However, the subsumption problem for their *union* $\mathcal{ALCF}^{+,o,\sqcup}$ is undecidable [3]. Recently, *fusions* (or independent joints) have been proposed as a more robust way of combining DLs [5, 4, 36]. But even fusions behave badly if the class of models is not closed under disjoint unions, which is the case when nominals or the negation of roles are required [5, 4] (or if we combine logics of time and space—while linear orders are natural models of time, their disjoint unions are certainly not).

In this situation, a natural question arises as to whether there exist at all sufficiently general and useful ways of combining representation and reasoning systems preserving their good computational properties. *The main aim of this paper is to give a positive answer to this question.* We propose a combination method which is robust in the computational sense and still allows for certain interactions between the combined components. Given n 'description systems' $L_1, \dots, L_n$ talking about domains $D_1, \dots, D_n$, we form a new language L containing the languages L_i, $1 \leq i \leq n$, and talking about the disjoint union $\bigcup_{i=1}^{n} D_i$, in which the D_i are connected by a finite number of relations $E_j \subseteq D_1 \times \dots \times D_n$, $1 \leq j \leq m$. The fragments L_i of L still talk about D_i; moreover, the super-language L contains $n \times m$ extra $(n-1)$-ary operators $\mathcal{E}_j^i$ which, given an input $(X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_n)$, for $X_\ell \subseteq D_\ell$, return the set of all $x \in D_i$ such that

$$\forall \ell \neq i \; \exists x_\ell \in X_\ell \; (x_1, \dots, x_{i-1}, x, x_{i+1}, \dots, x_n) \in E_j.$$

In other words, $\mathcal{E}_j^i(X_1, \dots, X_{i-1}, X_{i+1}, \dots X_n)$ is the value of the i-th factor of

$$(X_1 \times \dots \times X_{i-1} \times D_i \times X_{i+1} \times \dots \times X_n) \cap E_j.$$

We call L an *E-connection* of $L_1, \dots, L_n$.

This is a rough idea. To make it more precise, we use the notion of *abstract description system* (ADS, for short) introduced in [5, 4]. Basically, all description, modal, temporal, epistemic and similar logics (in particular, modal logics of space) can be conceived of as ADSs. For this reason, ADSs form a right level of abstraction to study combinations of knowledge representation formalisms such as DLs, temporal logics (TLs, for short), spatial logics (SLs), etc. The main technical result of this paper is the following theorem: *every E-connection of any finite number of decidable ADSs is decidable as well.*

Here are three simple examples of E-connections; in more detail they will be discussed in Section 4.

DL–SL: A DL L_1 (say, $\mathcal{ALC}$ or $\mathcal{SHIQ}$ [22]) talks about a domain D_1 of abstract objects. A spatial logic L_2 (say, qualitative $\mathsf{S4}_u$ [35, 12, 33, 16] or quantitative $\mathcal{MS}$ of [34, 24]) talks about a spatial domain D_2. An obvious E-connection is given by the relation $E \subseteq D_1 \times D_2$ defined by taking $(x, y) \in E$ iff y belongs to the spatial extension of x whenever x occupies some space. Then, given an L_1-concept C, say, river, the operator $\mathcal{E}^2(C)$ provides us with the spatial extension of all rivers. Conversely, given a spatial region X of L_2, say, the Alps, $\mathcal{E}^1(X)$ provides the concept comprising all objects the spatial extensions of which have a non-empty intersection with X. The concept country $\sqcap\, \mathcal{E}^1(X)$ would denote then the union of all alpine countries.

DL–TL: Now, let L_3 be a temporal logic (say, point-based PTL [15] or Halpern-Shoham's logic of intervals HS [20]) and let D_3 be a set of time points or, respectively, time intervals interpreting L_3. In this case, a natural relation $E \subseteq D_1 \times D_3$ is given by taking $(x, y) \in E$ iff y belongs to the life-span of x.

DL–SL–TL: Further, we can combine all L_1, L_2, L_3 above into a single formalism by defining a ternary relation $E \subseteq D_1 \times D_2 \times D_3$ such that $(x, y, z) \in E$ iff y belongs to the spatial extension of x at moment (interval) z.

2 Abstract description systems

An abstract description system consists of an abstract description language and a class of admissible models specifying the intended semantics.

Definition 1. An *abstract description language* (ADL) is determined by a countably infinite set $\mathcal{V}$ of *set variables*, a countably infinite set $\mathcal{X}$ of *object variables*, a (possibly infinite) sequence $(R_i)_{i \in \mathcal{R}}$ of *relation symbols* of arity m_i, $i \in \mathcal{R}$, and a (possibly infinite) sequence $(f_i)_{i \in \mathcal{I}}$ of *function symbols* of arity n_i, $i \in \mathcal{I}$. The *terms* t_j of the ADL are built as

follows:

$$t ::= x \mid \neg t \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid f_i(t_1, \ldots, t_{n_i}),$$

where $x \in \mathcal{V}$ and the Boolean operators $\neg$, $\wedge$, $\vee$ are different from all the f_i. The *term assertions* of the ADL are of the form

- $t_1 \sqsubseteq t_2$, where t_1 and t_2 are terms,

and the *object assertions* are

- $R_i(a_1, \ldots, a_{m_i})$, for $a_1, \ldots, a_{m_i} \in \mathcal{X}$ and $i \in \mathcal{R}$;
- $a : t$, for $a \in \mathcal{X}$ and t a term.

The sets of term and object assertions together form the set of the *ADL-assertions*.

Examples

(1) We remind the reader that $\mathcal{ALC}$-concept expressions C are composed from concept names by means of the operators $\sqcap$, $\neg$, $\forall R$, and $\exists R$, where R is a role name. The concept expressions of $\mathcal{ALC}$ (or any other standard description logic extending $\mathcal{ALC}$) can be regarded as terms $C^\sharp$ of an ADL $\mathcal{ALC}^\sharp$. Namely, with each concept name A we associate a set-variable $A^\sharp$, and with each role R we associate two unary function symbols $f_{\forall R}$ and $f_{\exists R}$. And then we put inductively:

$$\begin{aligned} (C \sqcap D)^\sharp &= C^\sharp \wedge D^\sharp \\ (\neg C)^\sharp &= \neg C^\sharp \\ (\forall R.C)^\sharp &= f_{\forall R}(C^\sharp) \\ (\exists R.C)^\sharp &= f_{\exists R}(C^\sharp) \end{aligned}$$

The object names of $\mathcal{ALC}$ are treated as object variables of $\mathcal{ALC}^\sharp$ and the role names as its binary relations. Thus, term assertions of $\mathcal{ALC}^\sharp$ correspond to general TBoxes, while object assertions correspond to ABoxes. (The connection between roles R and the function symbols $f_{\forall R}$ and $f_{\exists R}$ is fixed by choosing a proper class of admissible models; see below.) For transformations of more expressive DLs into ADLs consult [5, 4].

(2) The language of the logic $\mathsf{S4}_u$ (i.e., Lewis's modal system S4 with the universal modality), with topological spaces as the intended interpretations, consists of set variables $X_1, \ldots$ (in the modal context, propositional variables), the interior operator I (the necessity operator), the closure operator C (the possibility operator), the universal quantifier $\boxdot$ (the universal box), and the Booleans [35, 12, 33]. The corresponding ADL $\mathsf{S4}_u^\sharp$ would contain then the set variables $X_1^\sharp, \ldots$, the unary function symbols f_I, f_C, $f_{\boxdot}$, and the Booleans (but no relation symbols). Besides, according to the definition, $\mathsf{S4}_u^\sharp$ must contain a countably infinite set of object variables a_i. The translation $^\sharp$ of $\mathsf{S4}_u$-formulas into terms of $\mathsf{S4}_u^\sharp$ is obvious; for example,

$$(\Box\varphi)^\sharp = f_\Box(\varphi^\sharp), \text{ for } \Box \in \{I, C, \boxdot\}.$$

Note, however, that $\mathsf{S4}_u^\sharp$ allows for object assertions of the form $a_i : t^\sharp$ which have no analogs in $\mathsf{S4}_u$.

(3) The logic of metric spaces $\mathcal{MS}$ of [34, 24][1] consists of terms constructed from set variables X_i and nominals N_i using the Booleans and the operators $\mathsf{A}_{\leq r}$, $\mathsf{E}_{\leq r}$, $\mathsf{A}_{>r}$ and $\mathsf{E}_{>r}$, for $r \in \mathbb{Q}^+$. Intuitively, given a set X in a metric space, $\mathsf{E}_{\leq r}X$ is the set of all points in the space located at distance $\leq r$ from (at least one point in) X. A point is in $\mathsf{A}_{>r}X$ iff the whole complement of its r-neighborhood is in X. Again, terms t of $\mathcal{MS}$ can be translated into terms $t^\sharp$ of an ADL $\mathcal{MS}^\sharp$ by associating with every nominal N_i a 0-ary function symbol f_{N_i} and with the operators $\mathsf{A}_{\leq r}$, $\mathsf{E}_{\leq r}$, $\mathsf{A}_{>r}$, $\mathsf{E}_{>r}$ corresponding unary function symbols $f_{\mathsf{A}_{\leq r}}$, $f_{\mathsf{E}_{\leq r}}$, $f_{\mathsf{A}_{>r}}$, $f_{\mathsf{E}_{\leq r}}$, and then proceeding as above. The location variables of $\mathcal{MS}$ correspond to the object variables of $\mathcal{MS}^\sharp$.

(4) In the same way the propositional temporal language PTL, in which formulas are composed from propositional variables by means of the Booleans and the binary operators $\mathcal{U}$ ('until') and $\mathcal{S}$ ('since'), can be represented as an ADL $\mathsf{PTL}^\sharp$. In this case we associate with $\mathcal{U}$ and $\mathcal{S}$ *binary* function symbols $f_{\mathcal{U}}$ and $f_{\mathcal{S}}$. Note again that $\mathsf{PTL}^\sharp$ allows for object assertions $a_i : t^\sharp$ which have no analogs in PTL. However, since our intended flow of time is $\langle \mathbb{N}, < \rangle$, object variables can be represented as $p \wedge \neg(\mathsf{T}\mathcal{U}p) \wedge \neg(\mathsf{T}\mathcal{S}p)$. As in (2) and (3), $\mathsf{PTL}^\sharp$ contains no relation symbols.

The semantics of ADLs is defined via abstract description models.

Definition 2. An *abstract description model* (ADM) for an ADL $\mathcal{L}$ is a structure of the form

$$\mathfrak{A} = \langle W, \mathcal{V}^\mathfrak{A}, \mathcal{X}^\mathfrak{A}, F^\mathfrak{A}, R^\mathfrak{A} \rangle,$$

where $\mathcal{V}^\mathfrak{A} = (x^\mathfrak{A})_{x \in \mathcal{V}}$, $\mathcal{X}^\mathfrak{A} = (a^\mathfrak{A})_{a \in \mathcal{X}}$, $F^\mathfrak{A} = (f_i^\mathfrak{A})_{i \in \mathcal{I}}$, $R^\mathfrak{A} = (R_i^\mathfrak{A})_{i \in \mathcal{R}}$, W is a non-empty set, $x^\mathfrak{A} \subseteq W$, $a^\mathfrak{A} \in W$, each $f_i^\mathfrak{A}$ is a function mapping n_i-tuples $\langle X_1, \ldots, X_{n_i} \rangle$ of subsets of W to a subset of W, and the $R_i^\mathfrak{A}$ are m_i-ary relations on W. The *value* $t^\mathfrak{A} \subseteq W$ of an $\mathcal{L}$-term t in $\mathfrak{A}$ is defined inductively by taking

$$(t_1 \wedge t_2)^\mathfrak{A} = t_1^\mathfrak{A} \cap t_2^\mathfrak{A}, \ (t_1 \vee t_2)^\mathfrak{A} = t_1^\mathfrak{A} \cup t_2^\mathfrak{A},$$
$$(\neg t)^\mathfrak{A} = W \setminus (t)^\mathfrak{A}, \ (f_i(t_1, \ldots, t_k))^\mathfrak{A} = f_i^\mathfrak{A}(t_1^\mathfrak{A}, \ldots, t_k^\mathfrak{A}).$$

[1] The logic we consider here is called $\mathcal{MS}_2$ in [34] and $\mathcal{MS}^\sharp$ in [24].

The *truth-relation* $\mathfrak{A} \models \varphi$ for an $\mathcal{L}$-assertion φ is defined in the obvious way:

- $\mathfrak{A} \models R_i(a_1, \dots, a_{m_i})$ iff $R_i^{\mathfrak{A}}(a_1^{\mathfrak{A}}, \dots, a_{m_i}^{\mathfrak{A}})$,
- $\mathfrak{A} \models a : t$ iff $a^{\mathfrak{A}} \in t^{\mathfrak{A}}$,
- $\mathfrak{A} \models t_1 \sqsubseteq t_2$ iff $t_1^{\mathfrak{A}} \subseteq t_2^{\mathfrak{A}}$.

If $\mathfrak{A} \models \varphi$ holds, we say that φ is *satisfied* in $\mathfrak{A}$.

Definition 3. An *abstract description system* (ADS) is a pair $(\mathcal{L}, \mathcal{M})$, where $\mathcal{L}$ is an ADL and $\mathcal{M}$ is a class of ADMs for $\mathcal{L}$ that is closed under the following operation: if $\mathfrak{A} = \langle W, \mathcal{V}^{\mathfrak{A}}, \mathcal{X}^{\mathfrak{A}}, F^{\mathfrak{A}}, R^{\mathfrak{A}} \rangle$ is in $\mathcal{M}$, and $\mathcal{V}^{\mathfrak{A}'} = (x^{\mathfrak{A}'})_{x \in \mathcal{V}}$, $\mathcal{X}^{\mathfrak{A}'} = (a^{\mathfrak{A}'})_{a \in \mathcal{X}}$ are new assignments to set and object variables in W then $\mathfrak{A}' = \left\langle W, \mathcal{V}^{\mathfrak{A}'}, \mathcal{X}^{\mathfrak{A}'}, F^{\mathfrak{A}}, R^{\mathfrak{A}} \right\rangle \in \mathcal{M}$.

Let us now return to examples (1)–(4) above and supply the ADLs $\mathcal{ALC}^{\sharp}$, $\mathsf{S4}_u^{\sharp}$, $\mathcal{MS}^{\sharp}$ and $\mathsf{PTL}^{\sharp}$ with their intended ADMs.

Examples (cont.)

(1) For $\mathcal{ALC}^{\sharp}$, the class $\mathcal{M}$ of ADMs is defined as follows. For any $\mathcal{ALC}$-model

$$\mathcal{I} = \langle \Delta, A_1^{\mathcal{I}}, \dots, R_1^{\mathcal{I}}, \dots, a_1^{\mathcal{I}}, \dots \rangle,$$

$\mathcal{M}$ contains the model $\mathfrak{M} = \langle \Delta, \mathcal{V}^{\mathfrak{M}}, \mathcal{X}^{\mathfrak{M}}, F^{\mathfrak{M}}, R^{\mathfrak{M}} \rangle$, where F consists of the function symbols $f_{\forall R_i}$ and $f_{\exists R_i}$, and R is the set of all role names of $\mathcal{ALC}$:

- $(A^{\sharp})^{\mathfrak{M}} = A^{\mathcal{I}}$, for all concept names A;
- $a^{\mathfrak{M}} = a^{\mathcal{I}}$, for all object names a;
- $R_i^{\mathfrak{M}} = R_i^{\mathcal{I}}$, for all roles R_i;
- $f_{\forall R_i} X = \{d \in \Delta \mid \forall d' \in \Delta \ (dR_i^{\mathcal{I}} d' \to d' \in X)\}$, for all role names R_i;
- $f_{\exists R_i} X = \{d \in \Delta \mid \exists d' \in \Delta \ (dR_i^{\mathcal{I}} d' \wedge d' \in X)\}$, for all role names R_i.

(2) An $\mathsf{S4}_u$-model $\mathcal{I} = \langle T, \mathbb{I}, \mathbb{C}, X_1^{\mathcal{I}}, \dots \rangle$ consists of a topological space $\langle T, \mathbb{I} \rangle$, where $\mathbb{I}$ is an interior operator mapping subsets X of T to their interior $\mathbb{I}(X)$ and satisfying the equations $\mathbb{I}(X \cap Y) = \mathbb{I}(X) \cap \mathbb{I}(Y)$, $\mathbb{I}\mathbb{I}(X) = \mathbb{I}(X)$, $\mathbb{I}(X) \subseteq X$ and $\mathbb{I}(T) = T$ for all $X, Y \subseteq T$, $\mathbb{C}$ is the closure operator defined by $\mathbb{C}(X) = T - \mathbb{I}(T - X)$, and the $X_i^{\mathcal{I}}$ are subsets of T (interpreting the set variables of $\mathsf{S4}_u$). Of course, the operators I and C of $\mathsf{S4}_u$ are interpreted by $\mathbb{I}$ and $\mathbb{C}$, respectively. We define the class $\mathcal{M}$ of ADMs for $\mathsf{S4}_u^{\sharp}$ by taking, for every such $\mathsf{S4}_u$-model $\mathcal{I}$, the ADMs

$$\mathfrak{M} = \left\langle T, \mathcal{V}^{\mathfrak{M}}, \mathcal{X}^{\mathfrak{M}}, f_I^{\mathfrak{M}}, f_C^{\mathfrak{M}}, f_{\boxdot}^{\mathfrak{M}} \right\rangle,$$

where $(X_i^{\sharp})^{\mathfrak{M}} = X_i^{\mathcal{I}}$, $a^{\mathfrak{M}} \in \Delta$, for $a \in \mathcal{X}$, $f_I^{\mathfrak{M}} = \mathbb{I}$, $f_C^{\mathfrak{M}} = \mathbb{C}$ and, for every $Y \subseteq T$,

$$f_{\boxdot}^{\mathfrak{M}} Y = \begin{cases} \emptyset & \text{if } Y \neq T \\ T & \text{if } Y = T. \end{cases}$$

(3) An $\mathcal{MS}$-model $\mathcal{I} = \langle W, \delta, X_1^{\mathcal{I}}, \dots, N_1^{\mathcal{I}}, \dots a_1^{\mathcal{I}}, \dots \rangle$ consists of a metric space $\langle W, \delta \rangle$ together with interpretations of set variables X_i as subsets $X_i^{\mathcal{I}}$ of W, location variables a_i as elements $a_i^{\mathcal{I}}$ of W, and nominals N_i as singleton subsets $N_i^{\mathcal{I}}$ of W. Every such model $\mathcal{I}$ gives rise to the ADM

$$\mathfrak{M} = \langle W, \mathcal{V}^{\mathfrak{M}}, \mathcal{X}^{\mathfrak{M}}, F^{\mathfrak{M}} \rangle$$

for $\mathcal{MS}^{\sharp}$, where $(X_i^{\sharp})^{\mathfrak{M}} = X_i^{\mathcal{I}}$, $a_i^{\mathfrak{M}} = a_i^{\mathcal{I}}$, and $F^{\mathfrak{M}}$ consists of all functions f_{N_i}, $f_{\mathsf{A}_{\leq r}}$, $f_{\mathsf{E}_{\leq r}}$, $f_{\mathsf{A}_{>r}}$, $f_{\mathsf{E}_{>r}}$, for $r \in \mathbb{Q}^+$, defined by

$$f_{N_i}^{\mathfrak{M}} = N_i^{\mathcal{I}}$$
$$f_{\mathsf{A}_{\leq r}}^{\mathfrak{M}}(Y) = \{w \in W \mid \forall x \in W \ (\delta(w,x) \leq r \to x \in Y)\}$$
$$f_{\mathsf{E}_{\leq r}}^{\mathfrak{M}}(Y) = \{w \in W \mid \exists x \in W \ (\delta(w,x) \leq r \wedge x \in Y)\}$$
$$f_{\mathsf{A}_{>r}}^{\mathfrak{M}}(Y) = \{w \in W \mid \forall x \in W \ (\delta(w,x) > r \to x \in Y)\}$$
$$f_{\mathsf{E}_{>r}}^{\mathfrak{M}}(Y) = \{w \in W \mid \exists x \in W \ (\delta(w,x) > r \wedge x \in Y)\}$$

(4) The class of ADMs for $\mathsf{PTL}^{\sharp}$ consists of structures

$$\mathfrak{M} = \langle \mathbb{N}, \mathcal{V}^{\mathfrak{M}}, \mathcal{X}^{\mathfrak{M}}, F^{\mathfrak{M}} \rangle,$$

in which $F^{\mathfrak{M}}$ contains two binary functions defined by taking, for all $Y, Z \subseteq \mathbb{N}$,

$$f_U^{\mathfrak{M}}(Y, Z) = \{u \in \mathbb{N} \mid \exists z > u (z \in Z \wedge \forall y \in (u, z) y \in Y)\}$$
$$f_S^{\mathfrak{M}}(Y, Z) = \{u \in \mathbb{N} \mid \exists z < u (z \in Z \wedge \forall y \in (z, u) y \in Y)\},$$

where $(u, v) = \{w \in \mathbb{N} \mid u < w < v\}$.

The main reasoning task for ADSs we are concerned with is the *satisfiability problem* for finite sets of assertions. In DL, this corresponds to the satisfiability of ABoxes with respect to general TBoxes.

Definition 4. Let $\mathcal{S} = (\mathcal{L}, \mathcal{M})$ be an ADS. A finite set Γ of $\mathcal{L}$-assertions is called *satisfiable* in $\mathcal{S}$ if there is an ADM $\mathfrak{A} \in \mathcal{M}$ which satisfies all assertions in Γ.

The satisfiability problem for an ADS $\mathcal{S}$ restricted to sets Γ of *object assertions* will be called the *A-satisfiability problem* for $\mathcal{S}$.

The following theorem is an almost immediate consequence of the decidability of the corresponding logics, consult [17] for $\mathsf{S4}_u$, [34] for $\mathcal{MS}$, and [15] for PTL.

Theorem 5. *The satisfiability problem is decidable for the ADSs $\mathsf{S4}_u^{\sharp}$, $\mathcal{MS}^{\sharp}$, and $\mathsf{PTL}^{\sharp}$. It is also decidable for $\mathcal{L}^{\sharp}$ whenever $\mathcal{L}$ is a DL with the decidable satisfiability problem for ABoxes with respect to general TBoxes.*

3 Connections of ADSs

We are in a position now to define E-connections formally. Suppose that we have ADSs $\mathcal{S}_i = (\mathcal{L}_i, \mathcal{M}_i)$, for $1 \leq i \leq n$, with disjoint vocabularies apart from the Boolean operators.[2] Let E be a set of $n \times m$ function symbols $\mathcal{E}^i_j$ of arity $n-1$, for $1 \leq j \leq m$, $1 \leq i \leq n$. Define by induction the notions of *i-term* and *i-assertion* of the *E-connection* $\mathcal{C} = \mathcal{C}(\mathcal{S}_1, \dots, \mathcal{S}_n)$, for $1 \leq i \leq n$:

- every set variable of $\mathcal{L}_i$ is an i-term;
- the set of i-terms is closed under $\neg$, $\wedge$, $\vee$, and the non-Boolean function symbols of $\mathcal{L}_i$;
- if $(t_1, \dots, t_{i-1}, t_{i+1}, \dots, t_n)$ is a sequence of j-terms, for $j \neq i$, then $\mathcal{E}^i_k(t_1, \dots, t_{i-1}, t_{i+1}, \dots, t_n)$, for $1 \leq k \leq m$, are i-terms;
- if $(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n)$ is a sequence of object variables a_j from $\mathcal{L}_j$, for $j \neq i$, then the $\mathcal{E}^i_k(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n)$, $1 \leq k \leq m$, are i-terms.

The *i-term assertions* of $\mathcal{C}$ are of the form $t_1 \sqsubseteq t_2$, where t_1 and t_2 are i-terms. The *i-object assertions* are all expressions of the form $R_\ell(a_1, \dots, a_{m_\ell})$ and $a : t$, where R_ℓ is a relation symbol of $\mathcal{L}_i$, a and the a_k are object variables of $\mathcal{L}_i$, and t an i-term. The sets of all i-term and i-object assertions, $1 \leq i \leq n$, together form the set of *assertions* of the E-connection $\mathcal{C}$.

A structure of the form $\mathfrak{M} = \langle (\mathfrak{W}_i)_{i \leq n}, (E_j)_{j \leq m} \rangle$, where $\mathfrak{W}_i = \langle W_i, \mathcal{V}^{\mathfrak{W}}_i, \mathcal{X}^{\mathfrak{W}}_i, F^{\mathfrak{W}}_i, R^{\mathfrak{W}}_i \rangle \in \mathcal{M}_i$ and $E_j \subseteq W_1 \times \dots \times W_n$, is called a *model* for $\mathcal{C}$. The *extension* $t^{\mathfrak{W}} \subseteq W_i$ of an i-term t is defined by simultaneous induction. For set and object variables X and a of $\mathcal{L}_i$, we put $X^{\mathfrak{W}} = X^{\mathfrak{W}_i}$ and $a^{\mathfrak{W}} = a^{\mathfrak{W}_i}$. The inductive steps for the Booleans and function symbols of $\mathcal{L}_i$ are as before. The new clauses are:

$$(\mathcal{E}^i_j(t_1, \dots, t_{i-1}, t_{i+1}, \dots, t_n))^{\mathfrak{W}} =$$

$$\{x \in W_i \mid \exists \bigwedge_{\ell \neq i} x_\ell \in t^{\mathfrak{W}}_\ell \, (x_1, \dots, x_{i-1}, x, x_{i+1}, \dots, x_n) \in E_j\},$$

$$(\mathcal{E}^i_j(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n))^{\mathfrak{W}} =$$

$$\{x \in W_i \mid (a_1, \dots, a_{i-1}, x, a_{i+1}, \dots, a_n) \in E_j\}.$$

The *truth-relation* $\models$ between models $\mathfrak{M}$ for $\mathcal{C}$ and assertions of $\mathcal{C}$ is defined in the obvious manner:

- $\mathfrak{M} \models t_i \sqsubseteq t_2$ iff $t_1^{\mathfrak{W}} \subseteq t_2^{\mathfrak{W}}$;
- $\mathfrak{M} \models a : t$ iff $a^{\mathfrak{W}} \in t^{\mathfrak{W}}$;
- $\mathfrak{M} \models R_i(a_1, \dots, a_m)$ iff $R_i^{\mathfrak{W}}(a_1^{\mathfrak{W}}, \dots, a_{m_i}^{\mathfrak{W}})$.

A set Γ of assertions of $\mathcal{C}$ is called *satisfiable* if there is a model for $\mathcal{C}$ in which all assertions in Γ are true. And an assertion φ *follows* from a set of assertions Γ in $\mathcal{C}$ if $\mathfrak{M} \models \varphi$ whenever $\mathfrak{M} \models \Gamma$, for every model $\mathfrak{M}$ for $\mathcal{C}$. Note that the problem whether an assertion follows from a set of assertions can be reduced to the satisfiability problem. For example, $t_1 \sqsubseteq t_2$ follows from Γ iff $\Gamma \cup \{a : t_1 \wedge \neg t_2\}$, a a fresh object variable, is not satisfiable.

Note that, for any n object variables $a_i \in \mathcal{X}_i$ we have $\mathfrak{M} \models a_i : \mathcal{E}^i_j(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n)$ if and only if $E_j(a_1^{\mathfrak{W}}, \dots, a_n^{\mathfrak{W}})$. Hence, we can state that the tuple $\langle a_1^{\mathfrak{W}}, \dots, a_n^{\mathfrak{W}} \rangle$ is an instance of the relation E_j.

The main results of this paper are as follows:

Theorem 6. (i) *Suppose that the satisfiability problem for each of the ADSs $\mathcal{S}_i$, $1 \leq i \leq n$, is decidable. Then the satisfiability problem for any E-connection of the $\mathcal{S}_i$ is decidable as well.*

(ii) *If the $\mathcal{S}_i$ are decidable in EXPTIME, then the E-connection is decidable in 2EXPTIME.*

Corollary 7. *The satisfiability problem for any E-connection of DLs with decidable satisfiability problems for ABoxes with respect to general TBoxes and logics like* PTL, $\mathcal{MS}$, *and* $\mathsf{S4}_u$ *is decidable.*

We know of no example where the $\mathcal{S}_i$ are decidable in EXPTIME but the E-connection is 2EXPTIME-hard. It is an open problem whether such ADSs exist.

It is of interest that A-satisfiability is not preserved under E-connections. More precisely, the following 'negative' result holds, where $\mathcal{ALCF}$ is the extension of $\mathcal{ALC}$ with functional roles and the same-as constructor (it is known that A-satisfiability for $\mathcal{ALCF}$ is decidable [21, 27] while satisfiability is not [3]):

Theorem 8. *Let $\mathcal{S}$ be an arbitrary ADS. Then the A-satisfiability problem for any E-connection $\mathcal{C}(\mathcal{ALCF}, \mathcal{S})$ with a non-empty E is undecidable.*

Before presenting the proofs, we illustrate the notion of E-connection with illuminative examples.

4 Examples of E-connections

In this section we give four examples of E-connections using the representation formalisms introduced above. Our aim is to demonstrate the versatility of the combination technique and to outline its limits. The first three examples are 'two-dimensional,' while the fourth

[2] It is to be noted that this condition is different from the one required for fusions of ADSs in [5, 4]: when forming fusions, we assume that the set of set variables and object variables of the ADSs to be combined coincide. In the case of E-connections these sets of symbols should be disjoint, since they are used in the combined system to represent knowledge about disjoint domains.

one connects three ADSs. To simplify notation, we will not distinguish between description, spatial, metric, or temporal logics and the corresponding ADSs.

$\mathcal{C}(\mathcal{ALC}, \mathcal{MS})$: Suppose that you are developing a KR&R system for an estate agency. You imagine yourself to be a customer hunting for a house in London. What kind of requirements (constraints) could you have? Perhaps something like this:

(A) The house should not be too far from King's College, not more than 5 miles.

(B) The house should be close to shops, say, within 1 mile.

(C) There should be a 'green zone' around the house, at least within 2 miles in each direction.

(D) There must be a sports center around, and moreover, all sports centers of the district should be reachable on foot, i.e., they should be within, say, 3 miles.

(E) Public transport should easily be accessible: whenever you are not more than 8 miles away from home, the nearest bus stop or tube station should be reachable within 1 mile.

(F) The house should have a telephone.

(G) The neighbors shouldn't have children.

The terminology may require some clarification, so perhaps you may also need statements like

(H) All supermarkets are shops.

(Oh, you forgot about the most important constraint, the price, but let us deal with it a bit later.)

The resulting constraints (A)–(H) contain two kinds of knowledge. (F)–(H) can be classified as conceptual knowledge which is captured by almost any description logic, say, $\mathcal{ALC}$:

(F) $\mathit{house} : \exists \mathsf{has}.\mathsf{Telephone}$

(G) $\mathit{house} : \forall \mathsf{neighbor}.\forall \mathsf{child}.\bot$

(H) $\mathsf{Supermarket} \sqsubseteq \mathsf{Shop}$

(A)–(E) speak about distances and can be represented in the logic $\mathcal{MS}$ of metric spaces:

(A) $\mathit{house} \sqsubseteq \mathsf{E}_{\leq 5}\mathit{King's\_college}$

(B) $\mathit{house} \sqsubseteq \mathsf{E}_{\leq 1}\mathsf{Shop}$

(C) $\mathit{house} \sqsubseteq \mathsf{A}_{\leq 2}\mathsf{Green\_zone}$

(D) $\mathit{house} \sqsubseteq (\mathsf{E}_{\leq 3}\mathsf{Sports\_center}) \sqcap (\mathsf{A}_{>3}\neg\mathsf{Sports\_center})$

(E) $\mathit{house} \sqsubseteq \mathsf{A}_{\leq 8}\mathsf{E}_{\leq 1}\mathsf{Public\_transport}$

(Here, *house* and *King's_college* are nominals of $\mathcal{MS}$, while Shop, Green_zone, etc. are set variables.)

However, we can't just join these two knowledge bases together without connecting them. They speak about the same things, but from different points of view. For instance, in (H) 'shop' is used as a *concept*, while (B) deals with the *space* occupied by shops. Without connecting these different aspects we can't deduce from the knowledge base that a supermarket within 1 mile is sufficient to satisfy constraint (B). The required interaction can be easily captured by an E-connection between $\mathcal{ALC}$ and $\mathcal{MS}$. Indeed, take roles has, neighbor, child, concepts Telephone, Supermarket, Shop, Green_zone etc., and a nominal *King's_college* of $\mathcal{MS}$. Now, using the constructors $\mathcal{E}^2$ and $\mathcal{E}^1$ connecting $\mathcal{ALC}$- and $\mathcal{MS}$-models, we can represent constraints (A)–(H) as the concept Good_house defined by the following knowledge base in $\mathcal{C}(\mathcal{ALC}, \mathcal{MS})$:

$$\mathsf{Good\_house} = \mathsf{House} \sqcap \mathsf{Well\_located} \sqcap \exists \mathsf{has}.\mathsf{Telephone} \sqcap \forall \mathsf{neighbor}.\forall \mathsf{child}.\bot$$

$$\begin{aligned}\mathsf{Well\_located} = \mathcal{E}^1\Big(&\mathsf{E}_{\leq 5}\mathit{King's\_college} \sqcap \mathsf{A}_{\leq 2}\mathcal{E}^2(\mathsf{Green\_zone}) \\ &\sqcap \mathsf{E}_{\leq 1}\mathcal{E}^2(\mathsf{Shop}) \sqcap \mathsf{E}_{\leq 3}\mathcal{E}^2(\mathsf{Sports\_center}) \\ &\sqcap \mathsf{A}_{>3}\mathcal{E}^2(\neg\mathsf{Sports\_center}) \\ &\sqcap \mathsf{A}_{\leq 8}\mathsf{E}_{\leq 1}\mathcal{E}^2(\mathsf{Public\_transport})\Big)\end{aligned}$$

$$\mathsf{Supermarket} \sqsubseteq \mathsf{Shop}$$

If we want to specify also that the house should be of reasonable price, the ADS $\mathcal{ALC}^{\sharp}$ can be extended with a suitable concrete domain dealing with natural numbers so that the resulting ADS is still decidable [5, 4]. The E-connection will be decidable as well.

By using a satisfiability checking algorithm for satisfiability in arbitrary metric spaces we can verify only that the requirements are consistent. To answer the query whether such a house really exists in London, we need a suitable map of London as our metric space and the agency's knowledge base about properties.

$\mathcal{C}(\mathcal{ALCO}, \mathbf{S4}_u)$: Now imagine that you are employed by the EU parliament to develop a geographical information system about Europe. One part of the task is easy. You take the description logic $\mathcal{ALCO}$ (extending $\mathcal{ALC}$ with nominals, i.e., concept names which have to be interpreted as singleton sets [31, 23]) and, using concepts Country, Treaty, etc., nominals EU, Schengen_treaty, object names *Spain*, *Luxembourg*, *UK*, etc., and a role member, write

$\mathit{Luxembourg} : \exists \mathsf{member}.\mathsf{EU} \sqcap \exists \mathsf{member}.\mathsf{Schengen\_treaty}$

$\mathit{Iceland} : \exists \mathsf{member}.\mathsf{Schengen\_treaty} \sqcap \neg\exists \mathsf{member}.\mathsf{EU}$

$\mathit{France} : \mathsf{Country}$

$\mathsf{Schengen\_treaty} \sqsubseteq \mathsf{Treaty}$

$\exists \mathsf{member}.\mathsf{Schengen\_treaty} \sqsubseteq \mathsf{Country}$, etc.

After that you have to say something about the geography of Europe. To this end you can use the spatial

logic RCC-8 [29, 12, 30, 37] or the more expressive formalism of $\mathsf{S4}_u$ in which one can encode the topological meaning of the RCC-8 predicates—DC (disconnected), EQ (equal), EC (externally connected), NTPP (non-tangential proper part), etc.—as term assertions of $\mathsf{S4}_u$ (see e.g., [12, 37]), say,

$$\mathsf{DC}(X,Y) \text{ as } \neg\Diamond(X \wedge Y),$$
$$\mathsf{EQ}(X,Y) \text{ as } \boxdot(X \leftrightarrow Y),$$
$$\mathsf{EC}(X,Y) \text{ as } \Diamond(X \wedge Y) \wedge \neg\Diamond(IX \wedge IY),$$
$$\mathsf{NTPP}(X,Y) \text{ as } \boxdot(\neg X \vee IY) \wedge \Diamond(\neg X \wedge Y).$$

Using an E-connection between $\mathcal{ALCO}$ and $\mathsf{S4}_u$ you can then continue:

$$\mathsf{EQ}(\mathcal{E}^2(\mathsf{EU}), \mathcal{E}^2(\mathit{Portugal}) \sqcup \dots)$$
$$\mathsf{EC}(\mathcal{E}^2(\mathit{France}), \mathcal{E}^2(\mathit{Luxembourg}))$$
$$\mathsf{NTPP}(\mathcal{E}^2(\mathit{Luxembourg}), \mathcal{E}^2(\exists \mathsf{member.Schengen\_Treaty}))$$
$$\mathit{Austria} : \mathcal{E}^1(\mathit{alps})$$

i.e., 'the space occupied by the EU is the space occupied by its members,' 'France and Luxembourg have a common border,' 'if you cross the border of Luxembourg, then you enter a member of the Schengen Treaty,' 'Austria is an alpine country' (*alps* is a set variable of $\mathsf{S4}_u$). Of course, to ensure that the spatial extensions of the EU, *France*, etc. are not degenerate and to comply with requirements of RCC-8, you should guarantee that all mentioned spatial regions are interpreted by regular closed sets, i.e., $\mathcal{E}^2(\mathsf{EU}) = C\mathcal{I}\mathcal{E}^2(\mathsf{EU})$, $\mathcal{E}^2(\mathit{France}) = C\mathcal{I}\mathcal{E}^2(\mathit{France})$, etc.

Suppose now that you want to test your system and ask whether France is a member of the Schengen treaty, i.e., $\mathit{France} \sqsubseteq \exists \mathsf{member.Schengen\_treaty}$. The answer will be "Don't know!" because you did not tell your system that the spatial extensions of any two countries do not overlap. If you add, for example,

$$I\mathcal{E}^2(\mathsf{Country} \sqcap \neg\exists \mathsf{member.Schengen\_treaty}) \sqsubseteq \neg I(\mathcal{E}^2(\exists \mathsf{member.Schengen\_treaty}))$$

('the members of the Schengen treaty do not overlap with the non-Schengen countries') to the knowledge base, then the answer to the query will be "Yes!"

$\mathcal{C}(\mathcal{SHIQ}, \mathcal{ALCO})$: Having satisfied your boss in the EU parliament with the constructed GIS, you get a new task: develop a knowledge base regulating relations between people in the EU (citizenship, jobs, etc.). On the one hand, you already have the $\mathcal{ALCO}$ knowledge base describing countries in the EU from the previous example. But on the other hand, you must also be able to express laws like (i) 'no citizen of the EU may have more than one spouse,' (ii) 'all children of UK citizens are UK citizens,' or (iii) 'a person living in the UK is either child of somebody who is a UK citizen or has a work permit in the UK, or the person is a UK citizen or has a work permit in the UK herself.' This means, in particular, that you need more constructors than are provided by $\mathcal{ALCO}$, say, qualified number restrictions and inverse roles. It is known, however, that inverse roles, number restrictions, and nominals are difficult to handle algorithmically in one system [23]. The fusion of $\mathcal{ALCO}$ with, say, $\mathcal{SHIQ}$ of [22], having the required constructors, doesn't help either because transfer results for fusions are available so far only for DLs whose models are closed under disjoint unions [5, 4]—which is not the case if nominals are allowed as concept constructors. It seems that a perspective way to attack this problem is to *connect* $\mathcal{SHIQ}$ with $\mathcal{ALCO}$.

Let E contain three binary relations between the domains of $\mathcal{SHIQ}$ (people, companies, etc.) and $\mathcal{ALCO}$ (countries): $x\mathcal{S}y$ means that x is a citizen of y, $x\mathcal{L}y$ means that x lives in y, and $x\mathcal{W}y$ that x has a work permit in y. For example, $\mathcal{L}^1(\mathit{UK})$ denotes all people living in the UK, while $\mathcal{S}^1(\mathit{UK})$ all UK citizens. The subsumptions below represent the regulations (i)–(iii):

$$\mathcal{S}^1(\mathsf{EU}) \sqsubseteq \neg\exists_{\geq 2}\mathsf{married}.\top$$
$$\exists \mathsf{child\_of}.\mathcal{S}^1(\mathit{UK}) \sqsubseteq \mathcal{S}^1(\mathit{UK})$$
$$\mathcal{L}^1(\mathit{UK}) \sqsubseteq \exists \mathsf{child\_of}^{-1}.(\mathcal{S}^1(\mathit{UK}) \sqcup \mathcal{W}^1(\mathit{UK})) \sqcup \mathcal{S}^1(\mathit{UK}) \sqcup \mathcal{W}^1(\mathit{UK})$$

$\mathcal{C}(\mathcal{ALCO}, \mathsf{S4}_u, \mathsf{PTL})$ "The EU is developing," said your boss, "we are going to have new members by 2005." So you extend the connection $\mathcal{C}(\mathcal{ALCO}, \mathsf{S4}_u)$ with one more ADS—propositional temporal logic PTL. Now, besides nominals EU, etc. and object names *Germany*, etc. of $\mathcal{ALCO}$ and set variables *alps*, *Basel*, etc. of $\mathsf{S4}_u$, we have *nominals* $0, 1, \dots$ interpreting time points (n can be regarded as an abbreviation for $\neg \bigcirc_P^n \top \wedge \bigcirc_P^{n-1} \top$, where $\bigcirc_P\varphi$ stands for $\bot\mathcal{S}\varphi$). The ternary relation $\mathcal{E}(x, y, z)$ means now that at moment z (from the domain of PTL) point y (in the domain of $\mathsf{S4}_u$) belongs to the spatial region occupied by object x (in the domain of $\mathcal{ALCO}$). Then we can say, for example:

$$\mathcal{E}^2(\mathit{Poland}, 2005) \sqsubseteq \mathcal{E}^2(\mathsf{EU}, 2005)$$
$$\mathsf{PO}(\mathcal{E}^2(\mathit{Austria}, 1914), \mathcal{E}^2(\mathit{Italy}, 1950))$$
$$\Box_F\neg\mathcal{E}^3(\mathit{Basel}, \mathsf{EU}),$$

i.e., 'in 2005, the territory of Poland will belong to the territory occupied by the EU,' 'the territory of Austria

in 1914 partially overlaps the territory of Italy in 1950,' 'no part of Basel will ever belong to the EU.'

5 Proof

We prove Theorem 6 for the connection $\mathcal{C} = \mathcal{C}(\mathcal{S}_1, \mathcal{S}_2)$ between *two* ADSs $\mathcal{S}_i = (\mathcal{L}_i, \mathcal{M}_i)$, for $i = 1, 2$, by *one* relation E. The extra function symbols $\mathcal{E}^1$ and $\mathcal{E}^2$ of the connection are then unary. The general case of connections between n ADSs is treated analogously.

Let Γ be a finite set of assertion of $\mathcal{C}$. Denote by $ob_i(\Gamma)$ the set of object names from $\mathcal{L}_i$ which occur in Γ, $i = 1, 2$. Let $\overline{1} = 0$, $\overline{0} = 1$, and

$$o_i(\Gamma) = \{\mathcal{E}^i \neg \mathcal{E}^{\overline{i}} a \mid a \in ob_i(\Gamma)\}.$$

Note that $a^{\mathfrak{M}} \notin (\mathcal{E}^i \neg \mathcal{E}^{\overline{i}} a)^{\mathfrak{M}}$, for any model $\mathfrak{M}$ for $\mathcal{C}$ and object name a of $\mathcal{L}_i$. For every i-term t of the form $\mathcal{E}^i s$, where s is a term or an object name of $\mathcal{L}_{\overline{i}}$ occurring in $\Gamma' = \Gamma \cup o_1(\Gamma) \cup o_2(\Gamma)$, we introduce a fresh set variable x_t of $\mathcal{L}_i$. Given an i-term t, denote by $sur_i(t)$—the *surrogate of* t—the term which results from t by replacing all subterms $t' = \mathcal{E}^i s$ of t that are not within the scope of an $\mathcal{E}^i$ by $x_{t'}$. Clearly, $sur_i(t)$ belongs to the language $\mathcal{L}_i$.

Denote by Θ_i, $i = 1, 2$, the closure under negation of the set of i-terms which occur in Γ'. Without loss of generality we can identify $\neg\neg t$ with t. Thus, Θ_i is finite. The *i-consistency-set* $\mathcal{C}(\Theta_i)$ is defined as the set $\{t_c \mid c \subseteq \Theta_i\}$, where

$$t_c = \bigwedge\{\chi \mid \chi \in c\} \wedge \bigwedge\{\neg\chi \mid \chi \in \Theta_i \setminus c\}.$$

Sometimes we will identify $t \in \mathcal{C}(\Theta_i)$ with the set of its conjuncts; then $s \in t$ means that s is a conjunct of t. By $\top_i$ we denote $x^i \vee \neg x^i$, where x^i is a set variable from $\mathcal{L}_i$.

We are now ready to reduce the satisfiability in the connection $\mathcal{C}$ to the satisfiability problem in the components $(\mathcal{L}_1, \mathcal{M}_1)$ and $(\mathcal{L}_2, \mathcal{M}_2)$:

Theorem 9. *Γ is satisfiable iff there exist sets $\Delta_1 \subseteq \mathcal{C}(\Theta_2)$ and $\Delta_2 \subseteq \mathcal{C}(\Theta_2)$, a relation $e \subseteq \Delta_1 \times \Delta_2$, and for each $t \in \Delta_i$, $i = 1, 2$, a fresh object name a_t from $\mathcal{L}_i$ for which the following hold: there are functions σ_i from $ob_i(\Gamma)$ into Δ_i such that $\mathcal{E}^i \neg \mathcal{E}^{\overline{i}} a \notin \sigma_i(a)$, for any $a \in ob_i(\Gamma)$, the union Γ_i of the sets*

$$\{sur_i(\bigvee \Delta_i) = \top_i\}; \quad \{a_t : sur_i(t) \mid t \in \Delta_i\};$$

$$\{a : sur_i(\sigma_i(a)) \mid a \in ob_i(\Gamma)\};$$

$$\{(a : sur_i(t)) \mid (a : t) \in \Gamma \text{ an } i\text{-term assertion}\};$$

$$\{sur_i(t_1) \sqsubseteq sur_i(t_2) \mid t_1 \sqsubseteq t_2 \in \Gamma \text{ an } i\text{-term assertion}\};$$

$$\{R_j(a_1, \ldots, a_{m_j}) \in \Gamma \mid R_j(a_1, \ldots, a_{m_j}) \text{ an } i\text{-object assertion}\}$$

is $(\mathcal{L}_i, \mathcal{M}_i)$-satisfiable, and

1. *for all $\mathcal{E}^1 s \in \Theta_1$, s a 2-term, and $t \in \Delta_1$, we have $\mathcal{E}^1 s \in t$ iff there is $t' \in \Delta_2$ with $(t, t') \in e$ and $s \in t'$,*
2. *for all $\mathcal{E}^1 a \in \Theta_1$, $a \in ob_2(\Gamma)$, and $t \in \Delta_1$, we have $\mathcal{E}^1 a \in t$ iff $(t, \sigma_2(a)) \in e$,*
3. *for all $\mathcal{E}^2 s \in \Theta_2$, s a 1-term, and $t \in \Delta_2$, we have $\mathcal{E}^2 s \in t$ iff there is $t' \in \Delta_1$ with $(t', t) \in e$ and $s \in t'$,*
4. *for all $\mathcal{E}^2 a \in \Theta_2$, $a \in ob_2(\Gamma)$, and $t \in \Delta_2$, we have $\mathcal{E}^2 a \in t$ iff $(\sigma_1(a), t) \in e$.*

Proof ($\Rightarrow$) Suppose Γ is satisfiable in $\mathcal{C}$. Take a model $\mathfrak{M} = ((\mathfrak{W}_1, \mathfrak{W}_2), E)$ which satisfies Γ. Let, for $d \in W_i$,

$$t(d) = \bigwedge\{t \in \Theta_i \mid d \in t^{\mathfrak{M}}\}$$

and let $\Delta_i = \{t(d) | d \in W_i\}$. Take a fresh object name a_t from $\mathcal{L}_i$ for every $t \in \Delta_i$. Define, for $a \in ob_i(\Gamma)$,

$$\sigma_i(a) = \bigwedge\{t \in \Theta_i \mid a^{\mathfrak{M}} \in t^{\mathfrak{M}}\} \in \Delta_i$$

and define $e \subseteq \Delta_1 \times \Delta_2$ by taking $(t, t') \in e$ iff there are $d_1 \in W_1$ and $d_2 \in W_2$ such that $t = t(d_1)$, $t' = t(d_2)$, and $d_1 E d_2$. It remains to check that Δ_i, σ_i and e are as required.

First, we show that Γ_i is $(\mathcal{L}_i, \mathcal{M}_i)$-satisfiable. Take the model

$$\mathfrak{W}_i = \left\langle W_i, \mathcal{V}_i^{\mathfrak{W}_i}, \mathcal{X}_i^{\mathfrak{W}_i}, R_i^{\mathfrak{W}_i}, F_i^{\mathfrak{W}_i} \right\rangle.$$

$\mathfrak{W}_i$ is almost as required. We just have to give the appropriate values to the fresh set variables x_t and the fresh object names a_t. To this end put $x_t^{\mathfrak{W}_i'} = t^{\mathfrak{M}}$ for every $t = \mathcal{E}_i s$, $x^{\mathfrak{W}_i'} = x^{\mathfrak{W}_i}$ for the remaining variables, and $a_t^{\mathfrak{W}_i'} \in t^{\mathfrak{M}}$ for every $t \in \Delta_i$ and $a^{\mathfrak{W}_i'} = a^{\mathfrak{W}_i}$ for the remaining object names. Note that

$$\mathfrak{W}' = \left\langle W_i, \mathcal{V}_i^{\mathfrak{W}_i'}, \mathcal{X}_i^{\mathfrak{W}_i'}, R_i^{\mathfrak{W}_i}, F_i^{\mathfrak{W}_i} \right\rangle \in \mathcal{M}_i.$$

This follows from the closure conditions for the class $\mathcal{M}_i$. To prove that $\mathfrak{W}_i' \models \Gamma_i$, it is enough to show by induction that for all $d \in W_i$ and $s \in \Theta_i$, we have $d \in (sur_i(s))^{\mathfrak{W}_i'}$ iff $d \in s^{\mathfrak{M}}$. We leave this to the reader. Thus, the sets Γ_i are $(\mathcal{L}_i, \mathcal{M}_i)$-satisfiable.

Now we check that e satisfies conditions (1)–(4).

(1) Suppose $\mathcal{E}^1 s \in \Theta_1$, s is a 2-term and $t \in \Delta_1$. Assume first that $\mathcal{E}^1 s \in t$. Let $t = t(d)$ for some

$d \in W_1$. Then there is $d' \in W_2$ with dEd' and $d' \in s^{\mathfrak{M}}$. But then $s \in t(d')$ and $(t, t(d')) \in e$. Assume now that $(t, t') \in e$ and $s \in t'$. Then there are $d \in W_1$ and $d' \in W_2$ with $t = t(d)$ and $t' = t(d')$ and dEd'. We have $d' \in s^{\mathfrak{M}}$ and so $d \in (\mathcal{E}^1 s)^{\mathfrak{M}}$. Hence $\mathcal{E}^1 s \in t$.

(3) is proved in the same manner.

(2) Suppose $\mathcal{E}^1 a \in \Theta_1$ and $t \in \Delta_1$. Assume first that $\mathcal{E}^1 a \in t$. Then, for any $d \in W_1$ with $t(d) = t$ we have $dEa^{\mathfrak{M}}$. Hence $(t, t(a^{\mathfrak{M}})) \in e$ and so $(t, \sigma_2(a)) \in e$. Conversely, let $(t, \sigma_2(a)) \in e$. Then $\mathcal{E}^2 \neg \mathcal{E}^1 a \notin \sigma_2(a)$. By (3), $\neg \mathcal{E}^1 a \notin t$. Hence $\mathcal{E}^1 a \in t$.

(4) is proved in the same manner as (2).

($\Leftarrow$) Suppose that Δ_i, σ_i and e satisfying the conditions of the theorem are given. We construct a model satisfying Γ. To this end take $\mathfrak{W}_i \in \mathcal{M}_i$ satisfying Γ_i, $i = 1, 2$. Let, for $d \in W_i$,

$$t(d) = \bigwedge \{t \in \Theta_i \mid d \in (sur_i(t))^{\mathfrak{W}_i}\}.$$

Now define $E \subseteq W_1 \times W_2$ by taking dEd' iff $(t(d), t(d')) \in e$. We show that $\mathfrak{M} = (\mathfrak{W}_1, \mathfrak{W}_2, E)$ satisfies Γ. To this end we show by simultaneous induction for $i = 1, 2$ and all $d \in W_i$ and $s \in \Theta_i$, that $d \in (sur_i(s))^{\mathfrak{W}_i}$ iff $d \in s^{\mathfrak{M}}$. For set-variables the claim follows from the definition. Also the steps for the Boolean operators and for the function symbols of $\mathcal{L}_i$, $i = 1, 2$, are clear. It remains to consider the cases $t = \mathcal{E}^i s$, $i = 1, 2$. Let us assume $i = 1$, the case $i = 2$ is similar.

Suppose first that $t = \mathcal{E}^1 s$ for a 2-term s and that $d \in (sur_1(\mathcal{E}^1 s))^{\mathfrak{W}_1}$. Then $\mathcal{E}^1 s \in t(d)$. We know that

$$\mathfrak{W}_1 \models sur_1(\bigvee \Delta_1) = \top_1$$

and so $t(d) \in \Delta_1$. By (1), we find $t' \in \Delta_2$ with $(t(d), t') \in e$ and $s \in t'$. We know that

$$\mathfrak{W}_2 \models a_{t'} : sur_2(t'),$$

and so we find $d' \in W_2$ with $t' = t(d')$. Hence dEd'. From $s \in t'$ we obtain $d' \in (sur_2(s))^{\mathfrak{W}_2}$ and by the induction hypothesis $d' \in s^{\mathfrak{M}}$. Now $d \in (\mathcal{E}^1 s)^{\mathfrak{M}}$ follows. Conversely, suppose $d \in (\mathcal{E}^1 s)^{\mathfrak{M}}$. We find $d' \in W_2$ with dEd' and $d' \in s^{\mathfrak{M}}$. By the induction hypothesis, $d' \in (sur_2(s))^{\mathfrak{W}_2}$ and so $s \in t(d')$. By definition $(t(d), t(d')) \in e$ and so, by (1), $\mathcal{E}^1 s \in t(d)$ which implies $d \in (sur_1(\mathcal{E}^1 s))^{\mathfrak{W}_1}$.

Suppose now that $t = \mathcal{E}^1 a$ for an object name a of $\mathcal{L}_2$. Since $d \in (sur_1(\mathcal{E}^1 a))^{\mathfrak{W}_1}$, we have $\mathcal{E}^1 a \in t(d)$. As above we know that $t(d) \in \Delta_1$. By (2) $(t, \sigma_2(a)) \in e$. We also know that

$$\mathfrak{W}_2 \models a : sur_2(\sigma_2(a)).$$

Hence $dEa^{\mathfrak{W}_2}$, which implies $d \in (\mathcal{E}^1 a)^{\mathfrak{M}}$. Conversely, suppose that $d \in (\mathcal{E}^1 a)^{\mathfrak{M}}$. Then $dEa^{\mathfrak{M}}$, and so $(t(d), t(a^{\mathfrak{M}})) \in e$. We have $t(a^{\mathfrak{M}}) = \sigma_2(a)$ and therefore $(t(d), \sigma_2(a)) \in e$, which implies, by (2), that $\mathcal{E}^1 a \in t(d)$. This means $d \in (sur_1(\mathcal{E}^1 a))^{\mathfrak{W}_1}$. □

Theorem 6 follows from Theorem 9. Indeed, since the sets $\mathcal{C}(\Theta_i)$ are finite, Theorem 9 provides us with a decision procedure for $\mathcal{C}$ if decision procedures for $(\mathcal{L}_i, \mathcal{M}_i)$, $i = 1, 2$, are known. To decide whether a set Γ is satisfiable, 'guess' sets $\Delta_1 \subseteq \mathcal{C}(\Theta_1)$ and $\Delta_2 \subseteq \mathcal{C}(\Theta_2)$, functions $\sigma_i : ob_i(\Gamma) \to \Delta_i$, $i = 1, 2$, and a relation $e \subseteq \Delta_1 \times \Delta_2$ and check whether they satisfy the conditions listed in the formulation of the theorem. Regarding the complexity of the obtained decision procedure, the costly step is guessing the right sets Δ_i. The cardinality of the sets $\mathcal{C}(\Theta_i)$ is exponential in the size of Γ. Thus, there are double exponentially many different subsets to be chosen from. Since the cardinality of the chosen sets Δ_i may be exponential in the size of Γ, also the size of Γ_1 and Γ_2 may be exponential in Γ (because of the big disjunction over the Δ_i).

6 A-Satisfiability

Here we show that A-satisfiability is not preserved under connections. To this end, recall that $\mathcal{ALCF}$ is the extension of $\mathcal{ALC}$ by functional roles and the same-as constructor (see the introduction). A-satisfiability for $\mathcal{ALCF}$ is decidable [27], while satisfiability itself is not [3].

Proof of Theorem 8. Let $(\mathcal{L}, \mathcal{M})$ be an arbitrary ADS. We show that for any finite set Γ of general TBox-axioms and any assertion $a : t$ of $\mathcal{ALCF}$, there exists a set Γ^* of object assertions of $\mathcal{C} = \mathcal{C}(\mathcal{ALCF}, (\mathcal{L}, \mathcal{M}))$ such that $\Gamma \cup \{a : t\}$ is satisfiable iff Γ^* is satisfiable in the connection. Suppose Γ and $a : t$ are given. We may assume that Γ consists of axioms of the form $u = \top$. Denote by $\mathcal{R}$ the set of all roles which occur in $\Gamma \cup \{t\}$. We put $\mathcal{A}^i s = \neg \mathcal{E}^i \neg s$. Let b be an object name of $(\mathcal{L}, \mathcal{M})$.

Define Γ^* as the union of the sets

$$\{a : \mathcal{E}^1(b)\} \cup \{b : \mathcal{A}^2 f_{\forall R} \mathcal{E}^1(b) \mid R \in \mathcal{R}\},$$
$$\{a : t\} \cup \{b : \mathcal{A}^2 u \mid (u = \top) \in \Gamma\}.$$

Suppose $\Gamma \cup \{a : t\}$ is satisfied in an $\mathcal{ALCF}$-model $\mathfrak{W}_1$ with domain Δ. Define a model $\mathfrak{M}$ for $\mathcal{C}$ by taking an arbitrary model $\mathfrak{W}_2$ with domain W for $(\mathcal{L}, \mathcal{M})$ and putting $E = \Delta \times W$. It is easily checked that $\mathfrak{M} \models \Gamma^*$.

Conversely, let us suppose that $\mathfrak{M} \models \Gamma^*$ for a $\mathcal{C}$-model $\mathfrak{M} = (\mathfrak{W}_1, \mathfrak{W}_2, E)$. Let Δ be the domain of $\mathfrak{W}_1$. Denote by Δ' the minimal subset of Δ containing $a^{\mathfrak{M}}$

and satisfying the following closure condition for all $d, d' \in \Delta$:

$$\text{if } d \in \Delta' \text{ and } \exists S \in \mathcal{R}\ dS^{\mathfrak{M}}d' \text{ then } d' \in \Delta'.$$

The model $\mathfrak{M}_1'$, defined as the substructure of $\mathfrak{M}_1$ induced by Δ', satisfies $\Gamma \cup \{a : t\}$. To see this, it is sufficient to show that $u^{\mathfrak{M}} \supseteq \Delta'$ for every term u with $u = \top \in \Gamma$. Note that $d \in u^{\mathfrak{M}}$ whenever $(d, b^{\mathfrak{M}}) \in E$, because $b : \mathcal{A}^2 u \in \Gamma^*$. Hence it is enough to prove that for all $d \in \Delta'$, $(d, b^{\mathfrak{M}}) \in E$.

Since $a : \mathcal{E}^1(b) \in \Gamma^*$, $(a^{\mathfrak{M}}, b^{\mathfrak{M}}) \in E$. Suppose $d \in \Delta'$, dSd' for some $S \in \mathcal{R}$, and $(d, b^{\mathfrak{M}}) \in E$. We have $b \in \mathcal{A}^2 f_{\forall R}\mathcal{E}^1(b)$, and so $d \in f_{\forall R}\mathcal{E}^1(b)$. This implies $d' \in \mathcal{E}^1(b)$, from which $(d', b^{\mathfrak{M}}) \in E$. □

Note that in the proof we only used the following properties of $\mathcal{ALCF}$: (1) ABox-reasoning (without general TBox-axioms) is decidable, (2) reasoning with general TBoxes is undecidable, (3) an $\mathcal{ALCF}$-concept applies to an object d iff it applies to d in the generated substructure based on Δ' as defined above. Condition (3) applies to all standard description logics. This means that, roughly speaking, the A-satisfiability problem for an E-connection of two ADS is undecidable, whenever at least one ADS has an undecidable satisfiability problem.

7 Undefinability

Given that the E-connection of any finite number of decidable ADSs is decidable as well, it is clear that the interaction between the components has to be rather limited. In Section 4 we provided examples of potentially useful E-connections. Here we give some simple examples of what cannot be expressed by means of connections.

Definition 10. Let $\mathcal{C}$ be an E-connection. A property $\mathcal{P}$ of models of $\mathcal{C}$ is called definable in $\mathcal{C}$ if there exists a finite set Γ of assertions of $\mathcal{C}$ such that, for all models $\mathfrak{M}$ of $\mathcal{C}$, the following holds: $\mathfrak{M}$ has $\mathcal{P}$ iff $\mathfrak{M} \models \Gamma$.

The proof of the following theorem can be found in the full version of the paper.

Theorem 11. *Let R and S be role names.*

(i) *The property*

$$(\dagger) \quad \forall x \forall y \forall z\ (xRy \to (xEz \to yEz))$$

is not definable in the E-connections $\mathcal{C}(\mathcal{ALC}, \mathcal{MS})$, $\mathcal{C}(\mathcal{ALCO}, \mathsf{S4}_u)$ and $\mathcal{C}(\mathcal{SHIQ}, \mathcal{ALCO})$.

(ii) *The property*

$$(\ddagger) \quad \forall x \forall y\ (xRy \wedge xEx' \wedge yEy' \to x'Sy')$$

is not definable in the E-connection $\mathcal{C}(\mathcal{SHIQ}, \mathcal{ALCO})$.

For example, since we cannot express (†), we cannot say that the spatial extension of the capital of any country is included in the spatial extension of that country without enumerating the countries. Similarly, since (‡) is non-expressible, we cannot say that any child of any person who is a citizen of some country is a citizen of the same country.

8 Discussion

The investigation of combination methods for KR&R-formalisms consists, to a large extent, of the analysis of the trade-off between possible interactions of the components in the combined system and its computational properties. In this paper we studied a combination method which was proved to be extremely robust in the computational sense. Of course, the price for this is that the interaction between the components is limited. We hope, however, that starting from E-connections as a 'harmless' way of combining formalisms, it is possible to develop and study a hierarchy of more and more interactive combinations in a systematic manner.

Finally, note that the term 'harmless' used above is a bit misleading. Theorem 6 shows that decidability is inherited by the E-connection from its components. However, even if we have 'practical' algorithms for the components we do not obtain from our rather abstract model-theoretic proof a 'practical' algorithm for the E-connection. Moreover, it is unlikely that a 'practical' version of this general transfer result exists at all. Nevertheless, the proof of Theorem 6 indicates that in many cases existing practical decision procedures for the components can be combined so as to obtain practical decision procedures for the E-connection. We are currently working on such 'practical' combination techniques.

Acknowledgements

We should like to thank F. Baader and U. Sattler for helpful discussion. The work of M. Zakharyaschev was partially supported by U.K. EPSRC grants no. GR/R42474/01 and GR/R45369/01. The work of O. Kutz and F. Wolter was supported by DFG grant no. Wo 583/3-1.

References

[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983.

[2] A. Artale and E. Franconi. A temporal description logic for reasoning about actions and plans. *Journal of Artificial Intelligence Research*, 9:463–506, 1998.

[3] F. Baader, H.-J. Bürckert, B. Nebel, W. Nutt, and G. Smolka. On the expressivity of feature logics with negation, functional uncertainty, and sort equations. *Journal of Logic and Computation*, (2):1–18, 1993.

[4] F. Baader, C. Lutz, H. Sturm, and F. Wolter. Fusions of description logics and abstract description systems. To appear in Journal of Artificial Intelligence Research (JAIR) 2002. (Visit `www.informatik.uni-leipzig.de/~wolter`.)

[5] F. Baader, C. Lutz, H. Sturm, and F. Wolter. Fusions of description logics. In F. Baader and U. Sattler, editors, *2000 International Workshop on Description Logics*, Aachener Beiträge zur Informatik, pages 21–30. Wissenschaftsverlag Mainz in Aachen, 2000.

[6] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proceedings of the 13th IJCAI*, pages 446–451, Sidney, Australia, 1991.

[7] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proceedings of the 13th IJCAI*, pages 452–457, Sidney, Australia, 1991.

[8] F. Baader and P. Hanschke. Extensions of concept languages for a mechanical engineering application. In *Proceedings of the 16th German Annual Conference on Artificial Intelligence (GWAI'92)*, volume 671 of *Lecture Notes in Computer Science*, pages 132–143, Bonn (Germany), 1992. Springer.

[9] F. Baader and A. Laux. Terminological logics with modal operators. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 808–814, Montreal, Canada, 1995. Morgan Kaufmann, Los Altos.

[10] F. Baader and H.-J. Ohlbach. A multidimensional terminological knowledge representation language. *Journal of Applied Non-Classical Logics*, 5:153–197, 1995.

[11] F. Baader and W. Schulz, editors *Frontiers of Combinining Systems*. Kluwer Academic Publishers, 1996.

[12] B. Bennett. Modal logics for qualitative spatial reasoning. *Bulletin of the Interest Group in Pure and Applied Logic (IGPL)*, 4(1):23–45, 1996.

[13] C. Bettini. Time-dependent concepts: Representation and reasoning using temporal description logics. *Data and Knowledge Engineering*, 22:1–38, 1997.

[14] M. de Rijke and D. Gabbay, editors *Frontiers of Combining Systems II*. Research Studies Press LTD, England, 2000.

[15] D. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects*, volume 1. Oxford University Press, 1994.

[16] D. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*. Studies in Logic. Elsevier, 2002. (To appear; avaliable at `http://www.dcs.kcl.ac.uk/staff/mz`.)

[17] V. Goranko and D. Passy. Using the universal modality: gains and questions. *Journal of Logic and Computation*, 2:5–30, 1992.

[18] V. Haarslev, C. Lutz, and R. Möller. Defined topological relations in description logics. In *Proceedings of the 6th International Conference on the Principles of Knowledge Representation and Reasoning (KR'98)*, pages 112–124, 1998.

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

[20] J. Halpern and Y. Shoham. A propositional modal logic of time intervals. *Journal of the ACM*, 38:935–962, 1991.

[21] B. Hollunder and W. Nutt. Subsumption algorithms for concept languages. 1990. DFKI Research Report RR-90-04, Germany Research Center for Artificial Intelligence, Kaiserslautern.

[22] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proceedings of the Sixth International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, pages 161–180, 1999.

[23] I. Horrocks and U. Sattler. Ontology reasoning in the SHIQ(d) description logic. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 199–204. Morgan Kaufmann, Los Altos, 2001.

[24] O. Kutz, H. Sturm, N.-Y. Suzuki, F. Wolter, and M. Zakharyaschev. Logics of metric spaces. Submitted 2001. (Visit `http://www.informatik.uni-leipzig.de/~wolter`.)

[25] A. Laux. Beliefs in multi-agent worlds: A terminological approach. In *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI'94)*, pages 299–303, Amsterdam, The Netherlands, 1994.

[26] C. Lutz, H. Sturm, F. Wolter, and M. Zakharyaschev. Temporalizing tableaux. Manuscript, 2001.

[27] C. Lutz. Reasoning with concrete domains. In T. Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 90–95, Stockholm, Sweden, 1999. Morgan Kaufmann, Los Altos.

[28] C. Lutz. NEXPTIME-Complete Description Logics with Concrete Domains. In R. Gore, A. Leitsch and T. Nipkov, editors, *Automated Reasoning, Proceedings of the First International Joint Conference (IJCAR'2001)*, pages 45–60, Lecture Notes on Artificial Intelligence 2083, Springer, 2001.

[29] D. Randell, Z. Cui, and A. Cohn. A spatial logic based on regions and connection. In *Proceedings of the 3rd International Conference on the Principles of Knowledge Representation and Reasoning (KR'92)*, pages 165–176. Morgan Kaufmann, Los Altos, 1992.

[30] J. Renz. A canonical model of the Region Connection Calculus. In A. Cohn, L. Schubert, and S. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 6th International Conference (KR-98)*, pages 330–341. Morgan Kaufman, 1998.

[31] A. Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2):141–176, 1994.

[32] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the 13th IJCAI*, pages 466–471, Sidney, Australia, 1991.

[33] V. Shehtman. "Everywhere" and "Here". *Journal of Applied Non-Classical Logics*, 9, 1999.

[34] H. Sturm, N.-Y. Suzuki, F. Wolter, and M. Zakharyaschev. Semi-qualitative reasoning about distances: a preliminary report. In *Logics in Artificial Intelligence. Proceedings of JELIA 2000, Malaga, Spain*, pages 37–56, Berlin, 2000. Springer.

[35] A. Tarski. Der Aussagenkalkül und die Topologie. *Fundamenta Mathematicae*, 31:103–134, 1938.

[36] F. Wolter. Fusions of modal logics revisited. In *Advances in Modal Logic*, volume 1, pages 361–379, CSLI Publications, Stanford, 1998.

[37] F. Wolter and M. Zakharyaschev. Spatial reasoning in RCC-8 with Boolean region terms. In W. Horn, editor, *Proceedings of the 14th European Conference on Artificial Intelligence, ECAI 2000, Berlin, Germany*, pages 244–248. IOS Press, 2000.

[38] F. Wolter and M. Zakharyaschev. Spatio-temporal representation and reasoning based on RCC-8. In *Proceedings of the seventh Conference on Principles of Knowledge Representation and Reasoning, KR2000, Breckenridge, USA*, pages 3–14, Montreal, Canada, 2000. Morgan Kaufmann.

[39] F. Wolter and M. Zakharyaschev. Satisfiability problem in description logics with modal operators. In *Proceedings of the 6th International Conference on the Principles of Knowledge Representation and Reasoning (KR'98)*, pages 512–523. Morgan Kaufmann, Los Altos, 1998.

[40] F. Wolter and M. Zakharyaschev. Modal description logics: modalizing roles. *Fundamenta Informaticae*, 39:411–438, 1999.

[41] F. Wolter and M. Zakharyaschev. Multi-dimensional description logics. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 104–109. Morgan Kaufmann, Los Altos, 1999.

Evaluating a modular Abox algorithm

Sergio Tessaris, Ian Horrocks and Graham Gough
Department of Computer Science
University of Manchester
Manchester, UK
{tessaris|horrocks|gough}@cs.man.ac.uk

Abstract

This work constitutes an advance in the direction of the development of Description logic systems providing efficient and powerful reasoning in presence of individuals. We present the empirical evaluation of an algorithm for checking the satisfiability of Description logic knowledge bases.

The experiments we performed show that a modular algorithm, which separates terminological and hybrid reasoning, can provide as good performance as an hybrid Description logic reasoner. We are experimenting with the expressive Description logic $\mathcal{SHf}$, which extends the standard Description logic $\mathcal{ALC}$ with transitive roles, role hierarchy, and attributes.

1 Introduction

A description logic (DL) knowledge base (KB) is made up of two parts, a terminological part (the Tbox) and an assertional part (the Abox), each part consisting of a set of axioms. The Tbox asserts facts about *concepts* (sets of objects) and *roles* (binary relations), usually in the form of inclusion axioms, while the Abox asserts facts about *individuals* (single objects), usually in the form of instantiation axioms. For example, a Tbox might contain an axiom asserting that `Human` is subsumed by `Mammal`, while an Abox might contain axioms asserting that `John`, `Peter` and `Bill` are instances of the concept `Human` and that the pairs $\langle$`John`, `Peter`$\rangle$ and $\langle$`Peter`, `Bill`$\rangle$ are instances of the role `Brother`.

Transitive roles play an important part in knowledge representation, and they have been identified as a requirement in application domains concerned with complex physically composed objects (e.g. medical or engineering domains, see Artale et al. [1996], Rector and Horrocks [1997], Sattler [1995]). Several previous papers have also shown the efficacy of using a logic that includes a set of transitive roles, without incurring the cost entailed by the introduction of the transitive closure operator (see Horrocks and Gough [1997], Horrocks and Sattler [1998]).

There has been a great deal of work on the development of reasoning algorithms for expressive DLs (see Baader [1991], Horrocks and Sattler [1999], De Giacomo and Massacci [2000]), but most of this effort has been devoted to terminological reasoning (i.e. with an empty Abox). Important theoretical results have been published on the problem of reasoning with individuals as well as terminology (see for example Buchheit et al. [1993] or De Giacomo and Lenzerini [1996]). However, only a few works, like Horrocks et al. [2000] and the RACE system (see Haarslev and Möller [2000]), tackle the problem of developing practical Abox algorithms. This can be explained by the fact that for many applications Tbox reasoning can be enough, and realistic Aboxes tend to be far bigger than Tboxes. The size problem could lead to practical intractability, given the high complexity of reasoning in expressive DLs.

This paper presents an empirical evaluation of an algorithm which extends the above mentioned work for reasoning within a knowledge representation system which also includes ABox assertions. The approach used is based on an extension of the so–called *precompletion* technique (see Hollunder [1994], Donini et al. [1994]) for reducing the KB satisfiability problem to concept satisfiability. The cited works focused on DL knowledge bases with empty terminologies, and languages without transitive roles, while we generalise the precompletion technique for KBs with general inclusion axioms, transitive roles, and functional roles.

The main idea behind this technique consists of a correctness–preserving process which eliminates specific information regarding dependencies between individuals, while maintaining the consequences of such information. Once these dependencies are eliminated, the assertions

about a single individual can be independently verified, ignoring the fact that an individual is involved. The precompletion, or elimination of the dependencies, is performed by adding new assertions using a set of nondeterministic syntactic rules. Because of the nondeterminism of the rules, many different precompletions can be derived from a single knowledge base, which is satisfiable if and only if at least one of these precompletions is satisfiable.

The algorithm is completely independent from the terminological reasoner which is used for the concept satisfiability test. In this way, an Abox can be easily added to most existing DL systems without re-implementing the whole system. Moreover, optimisation strategies implemented at the terminological level do not adversely interact with the precompletion algorithm.

A further advantage of precompletion is that it confines the exponential blow up of the complexity to terminological reasoning. In fact, the size of precompletions is polynomial in the size of the KB. This may indicate that a system based on precompletion can run with a smaller memory footprint compared to a system based on an extension of Tableaux methods.[1] This, together with the fact that precompletion exploits the connectivity of the knowledge base, suggests that the algorithm can perform well in large and loosely connected KBs.

Due to space restrictions, correctness and completeness proofs for the algorithm are not included in this paper; they can be found in Tessaris [2001].

1.1 $\mathcal{SHf}$ knowledge bases

The DL $\mathcal{SHf}$ extends $\mathcal{ALC}$ with transitively closed primitive roles, role hierarchy and functional restrictions on roles. Valid concept expressions are defined by the syntax:

$$C ::= \top \mid \bot \mid A \mid \neg A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \forall R.C \mid \exists R.C$$

where A is a concept name chosen from a set $\mathcal{CN}$, and R is a role name chosen from a set $\mathcal{RN}$. We consider two disjoint subsets of $\mathcal{RN}$: the set of transitive role names $\mathcal{TRN}$, and the set of functional role names $\mathcal{FRN}$. In addition, we assume a set of individual names $\mathcal{O}$. A standard Tarski style model theoretic semantics provides the meaning for the expressions by means of interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ (see Schmidt-Schauss and Smolka [1991]). The set $\Delta^{\mathcal{I}}$ is the domain, and $\cdot^{\mathcal{I}}$ is an interpretation function which maps each concept name in $\mathcal{CN}$ to a subset of $\Delta^{\mathcal{I}}$, each role name in $\mathcal{RN}$ to a binary relation over $\Delta^{\mathcal{I}}$, and each individual name in $\mathcal{O}$ to a distinct element of $\Delta^{\mathcal{I}}$.[2] In addition, relations corresponding to transitive roles must be transitive, and the interpretation of functional roles must be a partial function. In the rest of the paper capital letters C or D indicate concepts, R or S roles, and small letters o, a or b individual names.

A DL knowledge base is a pair $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a *TBox* and $\mathcal{A}$ an *ABox*. The TBox, or terminology, contains concept axioms of the form $C \sqsubseteq D$ and role axioms of the form $S \sqsubseteq R$, while the ABox contains assertions about a set of individual names $\mathcal{O}$. These assertions are of the form $a{:}C$ or $\langle a, b \rangle{:}R$, where a,b are individual names in $\mathcal{O}$. We say that an interpretation satisfies an axiom $C \sqsubseteq D$ if the interpretation of D includes that of C, and analogously for inclusion between roles. An assertion $a{:}C$ is satisfied if the interpretation of a is an element of the interpretation of C; role assertions like $\langle a, b \rangle{:}R$ talk about the membership of the pair to the relation. A knowledge base Σ is said to be satisfiable if and only if there exists at least one interpretation which satisfies all the assertions in the knowledge base (i.e. the inclusion and membership relations represented in the KB are satisfied).

Role inclusion axioms contain only role names, and if there are cyclical definitions (e.g. $S \sqsubseteq R$ and $R \sqsubseteq S$), all the names involved in the cycle must correspond to the same binary relation in every interpretation (satisfying the axioms). For these reasons, we assume that role axioms are summarized by a partial order $\preceq$ defined over the set of role names.

2 KB satisfiability via precompletion

Let us consider for example a very simple Abox containing only the assertions:

$$\mathcal{A} = \{a{:}\forall R.C, \langle a, b \rangle{:}R, b{:}\neg C\}\,.$$

The two first assertions can be used to derive the new assertion $b{:}C$; it is easy to realise that $\mathcal{A}$ is satisfiable iff $\mathcal{A}' = \mathcal{A} \cup \{b{:}C\}$ is satisfiable as well (which is not the case, because b cannot be in C and $\neg C$ at the same time). The interesting point is that when we check the satisfiability of $\mathcal{A}'$ we do not need to consider the role assertion, because its effects have been made "explicit" in the new assertion. Therefore we can verify the KB satisfiability by checking the satisfiability of the concepts $\forall R.C$ and $C \sqcap \neg C$ separately.

Intuitively, the precompletion algorithm uses a set of transformation rules to propagate restrictions imposed by universal expressions along the role assertions in the Abox; in the example C is propagated to b because of $\forall R.C$. In the meantime, the propositional parts of the assertions (conjunctions and disjunctions) are expanded. Modal parts of the assertions (e.g. $\exists R.C$) are ignored and left to the terminological reasoner.

[1] This actually depends on the kind of optimisation implemented in the DL system. As we are going to suggest later on, the precompletion technique can be used as a testbed for improving strategies for Tableaux based systems.

[2] We use the unique name assumption for individuals.

In $\mathcal{SH}f$ the choice of whether an universal expression applies to a role assertion is complicated by the interaction between functional roles and role hierachy. For example let us consider a functional role F and two roles R, S included in F (i.e. $R \preceq F$ and $S \preceq F$). From the set of assertions $\{\langle a, b\rangle{:}R, a{:}\exists S.C\}$ we can derive $b{:}C$, because the functionality restriction on F forces any element related to a through R and S to be the same. This phenomenon is extended to "chains" of roles as well; e.g. like $R \preceq F_1$, $S \preceq F_1$, $S \preceq F_2$, and $T \preceq F_2$, with F_1 and F_2 functional roles. Precompletion rules are designed to take account of this interaction and to simplify their structure we introduce a set of binary operators $\cdot \approx_o \cdot$ (one for each individual name o in $\mathcal{O}$). Roughly speaking, two role names R and S are $\cdot \approx_o \cdot$ related if they are functional, and the Abox assertions force the R and S successors of the individual name o to be the same element (see Tessaris [2001] for its formal definition).

$\mathcal{A} \rightarrow_{\sqsubseteq} \{o{:}C\} \cup \mathcal{A}$
if o is in $\mathcal{O}$, $\top \sqsubseteq C$ is in $\mathcal{T}$
and $o{:}C$ is not in $\mathcal{A}$.

$\mathcal{A} \rightarrow_{\sqcup} \{o{:}D\} \cup \mathcal{A}$
if $o{:}C_1 \sqcup C_2$ is in $\mathcal{A}$,
and $D = C_1$ or $D = C_2$
and $o{:}D$ is not in $\mathcal{A}$.

$\mathcal{A} \rightarrow_{\exists^1} \{o'{:}C\} \cup \mathcal{A}$
if $o{:}\exists R.C$ and $\langle o, o'\rangle{:}S$ are in $\mathcal{A}$,
$R \approx_o S$, and $o'{:}C$ is not in $\mathcal{A}$.

$\mathcal{A} \rightarrow_{\forall^+} \{o'{:}\forall R.C\} \cup \mathcal{A}$
if $o{:}\forall T.C$ in $\mathcal{A}$, $\langle o, o'\rangle{:}S$ is in $\mathcal{A}$,
and there is $R \in \mathcal{TRN}$ such that $S \preceq R \preceq T$,
and $o'{:}\forall R.C$ is not in $\mathcal{A}$.

$\mathcal{A} \rightarrow_{\sqcap} \{o{:}C_1,\ o{:}C_2\} \cup \mathcal{A}$
if $o{:}C_1 \sqcap C_2$ is in $\mathcal{A}$,
and either $o{:}C_1$ or $o{:}C_2$ if not in $\mathcal{A}$.

$\mathcal{A} \rightarrow_{\forall^1} \{o'{:}C\} \cup \mathcal{A}$
if $o{:}\forall R.C$ and $\langle o, o'\rangle{:}S$ are in $\mathcal{A}$,
there is $R' \preceq R$ s.t. $R' \approx_o S$
and $o'{:}C$ is not in $\mathcal{A}$.

$\mathcal{A} \rightarrow_{\forall} \{o'{:}C\} \cup \mathcal{A}$
if $o{:}\forall R.C$ is in $\mathcal{A}$, and $\langle o, o'\rangle{:}S$ is in $\mathcal{A}$,
and $S \preceq R$, and $o'{:}C$ is not in $\mathcal{A}$.

Figure 1: Precompletion rules for $\mathcal{SH}f$

The input to the algorithm consists of a set containing the set of assertions in the Abox, which in this context are called *constraints*. The set of all concepts appearing in constraints like $o{:}C$, and associated to a single individual name is called the label of this name.

The rules in Figure 1 are repeatedly applied to the initial set until either no rule is applicable or a contradictory combination of constraints is detected (a so-called *clash*). The $\sqcup$-rule generates several alternative branches, which are exhaustively explored by means of *backtrack points* where the algorithm restarts in case of a clash detection.

Intuitively, a clash corresponds to a trivially unsatisfiable combination of constraints in the constraint set. When the precompletion process encounters a clash there is no need to continue adding new constraints, the precompletion will be unsatisfiable anyway; therefore the algorithm backtracks to the most recent nondeterministic choice and tries to explore the other possibilities. In $\mathcal{SH}f$ there are two kind of clashes. The first one occurs when two contradictory concepts are in the label of the same individual (e.g. a constraint system containing both $o{:}A$ and $o{:}\neg A$). The second one involves the functional roles, which may force two different individual names to be interpreted as the same element of the domain. This contradicts the unique name assumption, and the precompletion would be unsatisfiable. Due to the interaction between role hierarchy and functionality, the relation $\cdot \approx_o \cdot$ must be taken into account: two constraints $\langle o, a\rangle{:}R$, $\langle o, b\rangle{:}R'$ constitute a clash if $a \neq b$ and $R \approx_o R'$.

If none of the rules is applicable and there are no clashes, then a precompletion has been generated and must be checked for its satisfiability. This is done by gathering all the concept assertions concerning an individual. The concepts are then conjoined, generating a new single concept associated to each single individual. Each of these concepts is then verified by using an external call to a terminological reasoner; if they are all satisfiable then the algorithm terminates with success, otherwise it backtracks as in the case of clash detection. If the process fails to find a satisfiable precompletion after all the nondeterministic branches have been explored, then it terminates with failure.

As presented here the algorithm sounds rather naive and prone to inefficient exploration of the search space. In fact, we need a few "tricks" to avoid a hopelessly slow algorithm. In the following section we will show how to improve the algorithm to obtain acceptable behaviour in typical cases. Note that the theoretical worst case complexity of reasoning is EXPTIME-complete, therefore there can be pathological KBs manifesting this complexity.

3 Optimising the algorithm

The experience with previous DL systems shows that the direct implementation of the tableaux–based satisfiability algorithms provides very poor performance, unacceptable for any real application. Even though the experiments with other DL systems have been mainly at the terminological level, we can try to extract some lessons from those experiences (see Haarslev and Möller [1999], Horrocks and Patel-Schneider [1999]). The precompletion phase is completely separated from the terminological reasoning, so optimisation techniques implemented at the two levels do not

interact adversely. Moreover, the terminological reasoner we are using (FaCT) is already optimised, therefore we focus on the optimisation we can perform during the precompletion.

Section 3.1 introduces some of the well known optimization techniques adopted by most of the state of the art DL reasoners, while Section 3.2 and Section 3.3 describe techniques developed in conjunction with the precompletion algorithm.

3.1 Standard techniques

Axiom absorption and lazy expansion General axioms are one of the major sources of nondeterminism; in fact, in every axiom $C \sqsubseteq D$ is hidden the disjunctive formula $D \sqcup \neg C$ applied to every individual name. One of the most effective ways of reducing the effects of axioms is the so called *absorption* technique used in conjunction with lazy expansion of concept names (see Horrocks and Tobies [2000]).

Roughly speaking, the idea behind this technique is to transform a general axiom into the special form $A \sqsubseteq C$ where A is a concept name. Then the axiom is treated as a sort of definition for the name A and ignored until a concept constraint of the form $o{:}A$ is examined; at this point the "definition" C of A is added to the label of o (i.e. the new constraint $o{:}C$). This basic idea can be extended to negated concept names as well; i.e. having definitions of the form $\neg A \sqsubseteq C$. However, their combination must be used carefully to avoid incorrect results. We implemented the absorption algorithm described in Horrocks and Tobies [2000].

Lexical normalisation Concept expressions are normalised and encoded according to the transformation rules described in Horrocks and Patel-Schneider [1999]. In the normal form, concept expressions can be concept names, conjunctions of normal form concepts, universal quantification constructors, and the negation of a normal form. Conjunctions are represented as sets, so the order of the conjuncts does not affect the syntactic equivalence of different expressions; in addition, nested conjunctions are flattened (e.g. expressions like $((C_1 \sqcap C_2) \sqcap D)$ are transformed into $(C_1 \sqcap C_2 \sqcap D)$).

Backjumping Inherent unsatisfiability concealed in subproblems can lead to large amounts of unproductive backtracking search known as thrashing. Consider for example the set of constraints

$$\left\{ \begin{array}{l} a{:}(C_1 \sqcup D_1), \ldots, a{:}(C_n \sqcup D_n), a{:}\forall S.\forall R.\neg C, \\ b{:}\exists R.(C \sqcap D), \langle a, b\rangle{:}S \end{array} \right\},$$

which may cause the exploration of 2^n alternative combinations of constraints on a deriving from the concepts $(C_1 \sqcup D_1), \ldots, (C_n \sqcup D_n)$, while the true cause for the failure is related to the individual b. For example, this will happen if the rule application strategy forces the evaluation of all the constraints associated with an individual before considering a different individual.

This problem is addressed by adapting a form of dependency directed backtracking called *backjumping*, which has been used in solving constraint satisfiability problems (see Baker [1995]). Backjumping works by labeling concept constraints with a dependency set indicating the branching points on which they depend. A concept constraint $a{:}C$ depends on a branching point if $a{:}C$ was added to the label by the $\sqcup$–rule generating the branching point or if $a{:}C$ was generated by a different rule and the concept constraints involved in the rule depend on the branching point.[3]

When a clash is discovered, the dependency sets of the clashing concept constraints can be used to identify the most recent branching point where exploring the other branch might alleviate the cause of the clash. The algorithm can then jump back over intervening branching points without exploring alternative branches.

Note that when a contradiction is discovered by the verification of a label, there is no way for our algorithm to know the source of the contradiction; therefore, the union of the dependency set of all constraints of the label is returned. As we are going to show in Section 3.2, we can modify the algorithm in order to make the backjumping more effective in such cases.

With respect to the given example, the precompletion algorithm generates the first precompletion

$$\left\{ \begin{array}{l} a{:}(C_1 \sqcup D_1), \ldots, a{:}(C_n \sqcup D_n), a{:}\forall S.\forall R.\neg C, \\ a{:}C_1, \ldots, a{:}C_n \\ b{:}\exists R.(C \sqcap D), b{:}\forall R.\neg C, \langle a, b\rangle{:}S \end{array} \right\},$$

by choosing the first concept for each constraint containing the disjunction, and applying the $\forall$–rule to the constraints $a{:}\forall S.\forall R.\neg C$ and $\langle a, b\rangle{:}S$. In this process, the algorithm generates n different branching points, and every constraint $a{:}C_1, \ldots, a{:}C_n$ is labelled with a different branching point. The constraints associated with b have no branching point associated with them (this indicate a dependency with the top level), so when the terminological reasoner discovers the inconsistency of the label associated with b there is no reason to explore the alternative options for the individual a.

[3] Since new role assertions are never introduced, the dependency is only related to concept expressions.

3.2 Precompletion techniques

In the algorithm described above, the terminological reasoner is used only when a precompletion is generated. However, it can be used in different phases of the algorithm in a more sophisticated way.

Result caching As pointed out in Donini et al. [1996], caching the satisfiability results for the tested concept expressions is essential for maintaining the complexity of the actual algorithm in the EXPTIME theoretical complexity. In addition, in our case it can allow us to avoid the overhead of converting a concept expression from the internal representation to a format suitable for the terminological reasoner.

Early inconsistency detection This is based on the observation that if the constraints applied to an individual are inconsistent they will be inconsistent in any generated precompletion. Therefore the label of an individual can be verified even before an actual precompletion is generated. If an individual is found to have an inconsistent label, the algorithm stops exploring the current search branch and backtracks to the appropriate saved state of the precompletion process.

The label is verified when all the constraints associated to an individual have been considered, before considering a different individual. If during subsequent precompletion of the rest of the KB no new constraints are added to an individual which as been already precompleted, there is no reason to verify its label again.

Modal verification Backjumping only works properly with a precise knowledge of the constraints which generate a clash. Our problem is that in case of a clash detected by the terminological reasoner, we cannot get the information about the constraints responsible for the clash. The assumption that any concept in the label can be the cause of the clash makes the backjumping technique much less effective.

For example, the label $\{C, \neg D, \exists S.C, \exists R.C, \forall R.D\}$ is inconsistent if the terminology contains an axiom like $C \sqsubseteq \neg D$. However, the inconsistency can only be detected by the terminological reasoner. Once the unsatisfiability of the label is discovered, the system cannot tell which element of the label caused the inconsistency.

We can improve the algorithm by observing that if during the precompletion a clash has not been detected, the only possible cause for an inconsistency must be associated with an "anonymous" element whose existence is enforced by an existential quantification.[4] This is because the tree-like model property of the logic (see Vardi [1997], Tessaris [2001]) guarantees that no constraints can be "pushed back" from these "anonymous" elements, and contradictions generated in the propositional part of the labels (e.g. like C and $\neg C$) are detected during the precompletion process.

When there is the necessity of verifying the satisfiability of a label, the standard algorithm builds a new concept by conjoining all the concepts in the label. With the modal verification technique, the only concepts considered are the universal and existential quantifications (i.e. $\exists R.C$ and $\forall R.C$). Moreover, the algorithm selects the smallest subsets of the label which can be independently verified without compromising the completeness of the algorithm. In the previous example the terminological reasoner is used to verify the two concepts $(\exists R.C \sqcap \forall R.D)$ and $\exists S.C$ independently.

The basic idea for selecting these subsets is to consider each existential concept in a different subset, and adding the universal restrictions which can apply to it (i.e. having the same or a more general role name). However, this simple approach does not work with $\mathcal{SH}f$, because of the interaction between functional role and role hierachy. In fact, the operators $\cdot \approx_o \cdot$ introduced in Section 2 are used to group the relevant concepts of the label.

Using this technique, in case of unsatisfiability we can narrow the set of involved constraints. Therefore it can be extremely useful in conjunction with backjumping, because it enables a more precise identification of the backtrack point responsible for the clash.

Let us consider the following constraints

$$\left\{\begin{array}{l} a{:}(C_1 \sqcup C_2), \ldots, a{:}(C_{2n-1} \sqcup C_{2n}), \\ a{:}\forall R.\neg C, a{:}\exists R.(C \sqcap D) \end{array}\right\}.$$

In this case backjumping alone would not provide any improvement because the generated precompletions are of the form

$$\left\{\begin{array}{l} a{:}(C_1 \sqcup C_2), \ldots, a{:}(C_{2n-1} \sqcup C_{2n}), \\ a{:}C_{i_1}, \ldots, a{:}C_{i_n} \\ a{:}\forall R.\neg C, a{:}\exists R.(C \sqcap D) \end{array}\right\},$$

and the terminological reasoner is invoked to verify the concept

$$C_{i_1} \sqcap \ldots \sqcap C_{i_n} \sqcap \forall R.\neg C \sqcap \exists R.(C \sqcap D).$$

The algorithm is unable to detect which constraints are the cause of the unsatisfiability. By checking the modal part only, the terminological reasoner is invoked with the concept $\forall R.\neg C \sqcap \exists R.(C \sqcap D)$. When the unsatisfiability is reported, the algorithm knows that the clash is generated by one of the two constraints $a{:}\forall R.\neg C, a{:}\exists R.(C \sqcap D)$ and

[4] A constraint like $a{:}\exists R.C$ implies that there is an element related to individual name a through role R. The precompletion process would not investigate its existence since it is a task left to the terminological reasoner.

sibility to maximise the effect of optimisations like backjumping because it has better information about constraints causing contradictions (something we tried to simulate by verifying the modal part of the label only). We think that with classes like **k_branch_n** it is this fact that makes the real difference in performance. In addition, RACE is a more mature system and it has more optimisation techniques implemented (for example model merging, see Haarslev and Möller [2000]).

For these reasons, we did not expect to outperform RACE for any test having a small and almost fully connected Abox, where partitioning and memory occupation do not make any difference. Obviously, it could be the case that some of the results exhibited by our system depend on the peculiar structure of the test problems. However, if this behaviour is exhibited even with general KBs, it means that there is some unnecessary overhead in the hybrid reasoning in RACE which can be reduced by separating the treatment of Abox and "terminological" constraints. This would suggest that the performance of tableaux-based algorithms, such as the one implemented in RACE, might be improved by using an evaluation strategy more similar to the precompletion strategy (e.g. work on individuals first).

References

A. Artale, E. Franconi, N. Guarino, and L. Pazzi. Part-whole relations in object-centered systems: An overview. *Data and Knowledge Engineering (DKE)*, 20: 347–383, 1996.

F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 446–451. Morgan Kaufmann, 1991.

A. B. Baker. *Intelligent Backtracking on Constraint Satisfaction Problems: Experimental and Theoretical Results*. PhD thesis, University of Oregon, 1995.

M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

G. De Giacomo and M. Lenzerini. TBox and ABox reasoning in expressive description logics. In *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning (KR-96)*, pages 316–327. Morgan Kaufmann Publishers, 1996.

G. De Giacomo and F. Massacci. Combining deduction and model checking into tableaux and algorithms for Converse-PDL. *Information and Computation*, (162): 117–137, 2000.

F. M. 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.

F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: From subsumption to instance checking. *Journal of Logic and Computation*, 4 (4):423–452, 1994.

A. Era and F. Donini. Most specific concepts for knowledge bases with incomplete information. In *Int. Conf. on Information and Knowledge Management, Baltimore*, 1992.

E. Franconi, G. De Giacomo, R. M. MacGregor, W. Nutt, and C. A. Welty, editors. *Proceeding of the 1998 International Workshop on Description Logics (DL'98)*, 1998. CEUR Pubblication.

V. Haarslev and R. Möller. An empirical evaluation of optimization strategies for abox reasoning in expressive description logics. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. F. Patel-Schneider, editors, *Proceeding of the 1999 International Workshop on Description Logics (DL'99)*, pages 115–119. CEUR Pubblication, 1999.

V. Haarslev and R. Möller. Consistency testing: The RACE experience. In *Proceedings of TABLEAUX 2000*, volume 1847 of *Lecture Notes in Computer Science*, pages 57–61. Springer, 2000.

B. Hollunder. *Algorithmic Foundations of Terminological Knowledge Representation Systems*. PhD thesis, Universität des Saarlandes, 1994.

I. Horrocks and G. Gough. Description logics with transitive roles. In *Proceedings of the International Workshop on Description Logics, DL'97*. LRI, Universitè PARIS-SUD, Cente d'Orsay, 1997.

I. Horrocks and P. F. Patel-Schneider. DL system comparison. In Franconi et al. [1998].

I. Horrocks and P. F. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, 1999.

I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. In Franconi et al. [1998].

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, 1999.

I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In David

MacAllester, editor, *Proc. of the 17th Int. Conf. on Automated Deduction (CADE-17)*, number 1831 in Lecture Notes In Artificial Intelligence, pages 482–496. Springer-Verlag, 2000.

I. Horrocks and S. Tobies. Reasoning with axioms: Theory and practice. In *Proc. of the 7th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'2000)*, pages 285–296, 2000.

R. Küsters and R. Molitor. Approximating Most Specific Concepts in Description Logics with Existential Restrictions. In F. Baader, editor, *Proceedings of the Joint German/Austrian Conference on Artificial Intelligence, 24th German / 9th Austrian Conference on Artificial Intelligence (KI 2001)*, volume 2174 of *Lecture Notes in Artificial Intelligence*, Vienna, Austria, 2001. Springer–Verlag.

A. Rector and I. Horrocks. Experience building a large, reusable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*. AAAI Press, Menlo Park, California, 1997.

U. Sattler. A concept language for an engeneering application with part-whole relations. In A. Borgida, M. Lenzerini, D. Nardi, and B. Nebel, editors, *Proceedings of the International Workshop on Description Logics*, pages 119–123, Rome, 1995.

A. Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2):141–176, 1994.

M. Schmidt-Schauss and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.

S. Tessaris. *Questions and answers: reasoning and querying in Description Logic*. PhD thesis, University of Manchester, 2001.

M. Y. Vardi. Why is modal logic so robustly decidable? Technical Report TR97-274, Rice University, 1997.

Belief Revision

Resolving Inconsistencies by Variable Forgetting

Jérôme Lang
IRIT / Université Paul Sabatier
118 route de Narbonne
31602 Toulouse - France
lang@irit.fr

Pierre Marquis
CRIL / Université d'Artois
rue de l'Université - S.P. 16
62307 Lens - France
marquis@cril.univ-artois.fr

Abstract

In this paper, a fairly general framework for reasoning from inconsistent propositional belief bases is defined. Variable forgetting is used as a basic operation for weakening beliefs so as to restore consistency. The key notion is that of *recoveries*, which are sets of variables whose forgetting enables restoring consistency. Several criteria for defining preferred recoveries are proposed, depending on whether the focus is laid on the relative relevance of the variables or the relative entrenchment of the pieces of information (or both). Our framework encompasses several previous approaches as specific cases, including reasoning from preferred consistent subsets, and some forms of information merging and belief revision. Interestingly, the gain in flexibility and generality offered by our framework does not imply a complexity shift compared to these specific cases.

1 INTRODUCTION

Nontrivial processing of inconsistent sets of formulas is an important topic in artificial intelligence. Both the complexity of the problem and its significance are reflected by the number of approaches to paraconsistent reasoning that can be found in the literature under various names, like paraconsistent logics, belief revision, argumentative inference, information merging, model fitting, arbitration, knowledge integration, knowledge purification, etc.

Corresponding to these approaches are many different mechanisms to avoid trivialization. For instance, inconsistency can be removed by identifying wrong pieces of information through knowledge-gathering actions (see *e.g.* [Lang and Marquis, 2000] [Liberatore and Donini, 2000]). Inconsistency can also be dealt with. In this case, trivialization is avoided by weakening the set of consequences that can be derived from the belief base; this can be achieved either by considering an *approximation by below of classical entailment* (like in many paraconsistent logics, see [Besnard and Hunter, 1998] [Hunter, 1998] for a survey) or by *weakening the input belief base while keeping classical entailment*. This last technique is at work in so-called coherence-based approaches to paraconsistent inference (see *e.g.* [Rescher and Manor, 1970] [Fagin *et al.*, 1983] [Ginsberg, 1986] [Brewka, 1989], [Baral *et al.*, 1991] [Pinkas and Loui, 1992] [Benferhat *et al.*, 1993]), as well as in belief merging (see *e.g.* [Liberatore and Schaerf, 1998] [Revesz, 1997] [Konieczny and Pino Pérez, 1998] [Lin and Mendelzon, 1999]).

While some of these approaches take account for the relative importance of pieces of information, they do not handle the relative importance of *variables* in the problem at hand. This is problematic in many situations where some variables are less central than others, especially when some variables are meaningful only in the presence of others. Thus, it makes no sense to reason about whether John's car is grey if there is some strong conflict about whether John actually *has* a car.

In order to address this issue, we define in the following a new framework for reasoning from inconsistent propositional belief bases, using *variable forgetting* as a basic operation for weakening beliefs. Belief bases are viewed as (finite) vectors of formulas, conjunctively interpreted. W.l.o.g., each formula is assumed to be issued from a specific source of information. The key notion of our approach is that of *recoveries*, which are sets of variables whose forgetting enables restoring consistency. Several criteria for defining preferred recoveries are proposed, depending on whether the focus is laid on the relative relevance of the variables or the relative entrenchment of the pieces of information

(or both). Our framework offers several advantages compared with many existing approaches to paraconsistent reasoning. First of all, it is quite general and flexible. Especially, it enables to model situations where some sources of information are considered more reliable than others in an absolute way, but also relatively to some topics of interest. Equity between some sources of information can also be achieved by imposing to forget the same variables in the sources. Accordingly, our framework encompasses several previous approaches as specific cases, including reasoning from preferred consistent subsets, and forms of information merging and belief revision. Interestingly, the gain in generality and flexibility offered by our framework does not imply a complexity shift compared with these specific cases.

The rest of this paper is organized as follows. Formal preliminaries including a presentation of forgetting are given in Section 2. Our framework is presented in Section 3. Computational issues are discussed in Section 4. Section 5 concludes the paper. Proof sketches can be found in an appendix.

2 FORMAL PRELIMINARIES

2.1 PROPOSITIONAL LOGIC

$PROP_{PS}$ denotes the propositional language built up from a finite set PS of symbols, the Boolean constants $\top$ (true) and $\bot$ (false), and the usual connectives. $Var(\phi)$ denote the set of propositional variables occurring in the formula ϕ. $\phi_{x \leftarrow 0}$ (resp. $\phi_{x \leftarrow 1}$) denotes the formula obtained by replacing in ϕ every occurrence of variable x by $\bot$ (resp. $\top$).

An interpretation ω is an assignment of a truth value to each variable of PS. $\Omega = 2^{PS}$ is the set of all interpretations. Formulas are interpreted in the classical way. Every finite set of formulas is interpreted conjunctively. $Mod(\phi)$ denotes the set of models of ϕ. Finally, ω and ω' being two interpretations, $Diff(\omega, \omega')$ is the set of propositional variables assigned different truth values by ω and ω'.

2.2 FORGETTING

Our approach to restore consistency is based on *variable forgetting*, also known as projection or marginalization. Variable forgetting can be defined as follows (see [Lin and Reiter, 1994] [Lang and Marquis, 1998] for more details):

Definition 1 (forgetting) *Let ϕ be a formula from $PROP_{PS}$ and $V \subseteq PS$. The* forgetting *of V in ϕ, noted $\exists V.\phi$, is a quantified boolean formula over PS, equivalent to a formula from $PROP_{PS}$ that can be inductively defined as follows:*

- $\exists \emptyset.\phi \equiv \phi$;
- $\exists \{x\}.\phi \equiv \phi_{x \leftarrow 0} \vee \phi_{x \leftarrow 1}$;
- $\exists (\{x\} \cup V).\phi \equiv \exists V.(\exists \{x\}.\phi)$.

For example, $\exists \{a\}.\neg a \wedge b \equiv b$ and $\exists \{a\}.(a \vee b) \equiv \top$.

$\exists V.\phi$ represents the logically strongest consequence ψ of ϕ that is independent of V, which means that there exists a formula ψ' from $PROP_{PS}$ s.t. $\psi \equiv \psi'$ and $Var(\psi') \cap V = \emptyset$. Accordingly, forgetting a set of variables within a formula leads to *weaken* it. To be more precise, if $V \subseteq W$ holds, then $\exists V.\phi \models \exists W.\phi$ holds. Moreover, ϕ is consistent if and only if $\exists Var(\phi).\phi$ is valid (see [Lang and Marquis, 1998]).

Many characterizations of variable forgetting, together with complexity results, are reported in [Lang *et al.*, 2001]. For instance, for every $V \subseteq PS$ and every formula ϕ, we clearly have $\exists V.\phi \equiv \exists V_\phi.\phi$, where $V_\phi = V \cap Var(\phi)$.

3 REASONING FROM PREFERRED RECOVERIES

3.1 BELIEF BASES AND RECOVERIES

Let us first make precise the belief bases we are interested in.

Definition 2 (belief base) *A* belief base *B is a vector $\langle \phi_1, \ldots, \phi_n \rangle$ of n formulas from $PROP_{PS}$, where n is a positive integer.*

$B = \langle \phi_1, \ldots, \phi_n \rangle$ is conjunctively interpreted, so that it is said to be *inconsistent* if and only if $\{\phi_1, \ldots, \phi_n\}$ is inconsistent. Each i $(1 \leq i \leq n)$ identifies a source of information and ϕ_i denotes the piece of belief conveyed by source i. Note that it can be the case that a formula occurs more than once in B, which can be used to model the situation where two different sources (or more) give the same information.

The key notions of our approach are *forgetting vectors* and *recoveries*. A forgetting vector consists of a collection of sets of variables to be forgotten in each formula from the belief base. These sets of variables need not be identical, but they should obey some constraints bearing on the forgetting process:

Definition 3 (forgetting context) *A* forgetting context *$\mathcal{C}$ for a belief base $B = \langle \phi_1, \ldots, \phi_n \rangle$ is a triple $\langle F, G, H \rangle$ where:*

- $F = \langle F_i \rangle_{i=1..n}$ *where, for each* i, $F_i \subseteq PS$ *is the set of variables that can be forgotten in* ϕ_i *(*i.e., *variables in* $PS \setminus F_i$ *must not be forgotten in* ϕ_i*);*
- $G = \langle G_i \rangle_{i=1..n}$ *where, for each* i, G_i *is a directed graph on* F_i*; an arc* (v, v') *in* G_i *means that* v' *is meaningful only in the presence of* v, i.e., *that if* v *is forgotten in* ϕ_i *then* v' *must be forgotten as well in* ϕ_i*;*
- $H \subseteq PS \times \{1 \ldots n\} \times \{1 \ldots n\}$ *is a set of triples* (v, i, j) *where* $v \in F_i \cap F_j$ *and* $i \neq j$*, meaning that if* v *is forgotten in* ϕ_i*, then it must be forgotten as well in* ϕ_j.

F, G and H impose some constraints over the way the pieces of belief ϕ_i from B can be weakened. Thus, if $x \notin F_i$, forgetting x in ϕ_i is forbidden. This is helpful to model the situation where source i is fully reliable to what concerns x, *i.e.*, what i says about x must be taken for sure. G_i enables to express that variables can be significant only in presence of others so forgetting the latter imposes to forget the former. Finally, H can be used to force some sources of information to be considered on equal terms w.r.t. weakening. For instance, if ϕ_i and ϕ_j are together inconsistent and consistency can be recovered by forgetting x in ϕ_i, then it could be expected that x should also be forgotten in ϕ_j.

Example 1 *(inspired from [Konieczny and Pino Pérez, 1998]). As a matter of illustration, let us consider the following preference merging scenario. Suppose that a group of four co-owners of a residence tries to agree about building a new tennis court* t *and/or a new swimming pool* s*. If it is constructed, the swimming pool can be either red (*s_r*) or blue (*s_b*). If both a tennis court and a pool – respectively, one of them, none of them – are (is) constructed, the induced cost is 2 money units (*c_2*) – respectively, 1 unit (*c_1*), nothing* c_0*). The first co-owner would not like to spend more than 1 unit, and prefers a red swimming pool, should it be constructed. The second co-owner would like to improve the quality of the residence by the construction of at least a tennis court or a swimming pool and would prefer a blue swimming pool, should it be constructed. The third co-owner just prefers a swimming pool to be built, whatever its colour. The fourth co-owner would like both a swimming pool (of whatever colour) and a tennis court to be built.*

Of course, if there is no agreement about whether the swimming pool is to be constructed, any preference concerning its colour must be ignored. Furthermore, it is meaningless forgetting about whether the pool should be blue without forgetting whether it should be red: either we forget about the colour of the swimming pool or not. Similarly, either we forget about the expenses, i.e., we forget $\{c_0, c_1, c_2\}$ *or not (i.e., we forget none of them).*

Clearly enough, the preferences of the group are jointly inconsistent. This scenario can be encoded in our framework using the following belief base $B = \langle \phi_1, \phi_2, \phi_3, \phi_4, \phi_5 \rangle$ *and forgetting context* C*:*

- $\phi_1 = (s \Rightarrow (s_r \oplus s_b)) \wedge (\neg s \Rightarrow \neg s_r \wedge \neg s_b) \wedge (c_2 \Leftrightarrow (s \wedge t)) \wedge (c_1 \Leftrightarrow (s \oplus t)) \wedge (c_0 \Leftrightarrow (\neg s \wedge \neg t))$
- $\phi_2 = (c_0 \vee c_1) \wedge (s \Rightarrow s_r)$;
- $\phi_3 = (s \vee t) \wedge (s \Rightarrow s_b)$;
- $\phi_4 = s$;
- $\phi_5 = s \wedge t$;
- $F_1 = \emptyset$, $F_2 = F_3 = F_4 = F_5 = \{s, t, s_r, s_b, c_0, c_1, c_2\}$;
- $G_1 = \emptyset$, $G_2 = G_3 = G_4 = G_5 = \{(s, s_r), (s, s_b), (s_r, s_b), (s_b, s_r), (c_0, c_1), (c_0, c_2), (c_1, c_0), (c_1, c_2), (c_2, c_0), (c_2, c_1)\}$;
- $H = \emptyset$.

Given that $F_1 = \emptyset$, ϕ_1 *expresses the integrity constraint according to which the swimming pool must be either red or blue if it is constructed, as well as the logical definition of the induced price from the number of equipments built.* ϕ_2 *(resp.* ϕ_3, ϕ_4, ϕ_5*) encodes the preferences of the co-owners.* G_2 $(= G_3 = G_4 = G_5)$ *ensures that* s_r *and* s_b *are irrelevant if there is no agreement on* s*, that* s_r *is irrelevant if and only if* s_b *is, and that for any* $i, j \in \{0, 1, 2\}$, c_i *is irrelevant if and only if* c_j *is. In the situation where all co-owners must be considered on equal terms w.r.t. the set of variables to be forgotten,* H *must be changed to* $(\{s_r, s_b\} \times \{2, 3, 4, 5\} \times \{2, 3, 4, 5\}) \cup (\{c_0, c_1, c_2\} \times \{2, 3, 4, 5\} \times \{2, 3, 4, 5\})$.

As expected, our definition is very general and many different forgetting contexts can be considered. In some situations, the variables forgotten in each of the pieces of information must be identical so that all sources of information are considered on equal terms. This can be captured by considering *homogeneous* contexts, where a forgetting context $C = \langle F, G, H \rangle$ for $B = \langle \phi_1, \ldots, \phi_n \rangle$ is said to be homogeneous if and only if for every i, $j \in 1 \ldots n$, we have $F_i = F_j$ and $G_i = G_j$, and $H = PS \times \{1 \ldots n\} \times \{1 \ldots n\}$. A simple case when C should not be homogeneous is when some of the pieces of information must be left intact: for such a ϕ_i, we set $F_i = \emptyset$. This is useful to encode

integrity constraints (or revision formulas), *i.e.*, formulas that are required to be true (just like ϕ_1 in the example above). Another non-homogeneous forgetting context for $B = \langle \phi_1, \ldots, \phi_n \rangle$ is its *standard* forgetting context, noted $\mathcal{C}_\mathcal{S}$ and given by $F_i = PS$, $G_i = \emptyset$ for every $i \in 1 \ldots n$, and $H = \emptyset$. In this context, every variable can be forgotten in any piece of belief, and variables can always be forgotten in an independent way.

We can now define our notion of recovery:

Definition 4 (recovery) *Let $B = \langle \phi_1, \ldots, \phi_n \rangle$ be a belief base and $\mathcal{C}$ be a forgetting context for B. A* recovery vector *(or recovery for short) $\vec{V}$ for B given $\mathcal{C}$ is a vector $\vec{V} = \langle V_1, \ldots, V_n \rangle$ of subsets of $Var(B)$[1] s.t.:*

- *for every $i \in 1 \ldots n$, $V_i \subseteq F_i$, and*
- *for every $i \in 1 \ldots n$, for any $(v, v') \in G_i$, $v \in V_i$ implies $v' \in V_i$, and*
- *for every $(v, i, j) \in H$, $v \in V_i$ implies $v \in V_j$, and finally*
- *the set $B \mid \vec{V} = \{\exists V_i.\phi_i \mid i \in 1 \ldots n\}$ is consistent.*

$B \mid \vec{V}$ is called the projection *of recovery $\vec{V}$ on B given $\mathcal{C}$. $\mathcal{R}_\mathcal{C}(B)$ denotes the set of all recoveries for B given $\mathcal{C}$. B is said to be* recoverable *w.r.t. $\mathcal{C}$ if and only if $\mathcal{R}_\mathcal{C}(B) \neq \emptyset$.*

Example 1 (continued) *Here is a list of recoveries for B given $\mathcal{C}$. For each recovery $\vec{V}$ and every $i \in 1 \ldots 5$ we compute $\exists V_i.\phi_i$, and finally $B|\vec{V}$.*

All the vectors $\vec{V}^1$ to $\vec{V}^9$ considered on the table 1 are recoveries for B given $\mathcal{C}$, while the following ones are not *recoveries for B given $\mathcal{C}$:*

- *$\langle \emptyset, \{t\}, \{t\}, \{t\}, \{t\} \rangle$, because $\exists V_1.\phi_1 \wedge \exists V_2.\phi_2 \wedge \exists V_3.\phi_3 \wedge \exists V_4.\phi_4 \wedge \exists V_5.\phi_5$ is inconsistent;*
- *$\langle \emptyset, \{s\}, \{s\}, \{s\}, \{s\} \rangle$, because it does not satisfy the constraints of $\mathcal{C}$.*

By construction, replacing the beliefs of B by the projection on B of any of its recoveries is sufficient to restore consistency, provided that B is recoverable. In the general case, it may happen that $\mathcal{R}_\mathcal{C}(B)$ is empty, because some variables must not be forgotten. Another important sufficient condition for a belief base not to be recoverable is when it incorporates some inconsistent ϕ_i. In some situations, this can reflect the fact that the i^{th} source of information is fully unreliable, so it should not be taken into account at all (hence the corresponding ϕ_i should be removed from B).

[1] We note $Var(\langle \phi_1, \ldots, \phi_n \rangle) = \bigcup_{i=1}^{n} Var(\phi_i)$.

3.2 PREFERRED RECOVERIES

Generally, many recoveries for a belief base given a forgetting context are possible, but most of the time, some of the corresponding projections are more expected than others. Discriminating them calls for a notion of *preferred recovery*. Many preference criteria can be considered so as to capture in formal terms several intuitions about the way the pieces of information should be merged.

For instance, we may prefer the recoveries $\vec{V} = \langle V_1, \ldots, V_n \rangle$ for $B = \langle \phi_1, \ldots, \phi_n \rangle$ in which the set V_i of variables forgotten in some of the ϕ_i's are as close to each other as possible[2], and eventually coincide (which always is the case whenever homogeneous contexts are considered). We may also prefer recoveries that lead to forget minimal sets of variables, where minimality may be defined using a preference criterion induced by some priorities or penalties on the set of variables. Finally, we may prefer recoveries that preserve as much as preferred beliefs as possible (like in the coherence-based approaches to paraconsistent reasoning).

In order to capture these various preferences in formal terms, $\mathcal{R}_\mathcal{C}(B)$ can be equipped with a preordering (*i.e.*, a reflexive and transitive relation) $\sqsubseteq$. For every vector $\vec{V}$, $\vec{V'}$ of length n, $\vec{V} \subseteq_p \vec{V'}$ means that $V_i \subseteq V'_i$ for each $i \in 1 \ldots n$.

Definition 5 (preference relation) *Given a belief base $B = \langle \phi_1, \ldots, \phi_n \rangle$ and a forgetting context $\mathcal{C}$ for it, a* preference relation *is a reflexive and transitive relation $\sqsubseteq$ on $\mathcal{R}_\mathcal{C}(B)$ satisfying the monotonicity property:*

$$\forall \vec{V}, \vec{V'} \in \mathcal{R}_\mathcal{C}(B), \text{ if } \vec{V} \subseteq_p \vec{V'} \text{ then } \vec{V} \sqsubseteq \vec{V'}.$$

We note $\vec{V} \sqsubset \vec{V'}$ for $\vec{V} \sqsubseteq \vec{V'}$ and $\vec{V'} \not\sqsubseteq \vec{V}$, and $\vec{V} \sim \vec{V'}$ for $\vec{V} \sqsubseteq \vec{V'}$ and $\vec{V'} \sqsubseteq \vec{V}$.

It is easy to prove that for any $\vec{V}$, $\vec{V'} \in \mathcal{R}_\mathcal{C}(B)$, if $\vec{V} \subseteq_p \vec{V'}$ then $B \mid \vec{V} \models B \mid \vec{V'}$ holds. Accordingly, among the recoveries from $\mathcal{R}_\mathcal{C}(B)$, the minimal ones w.r.t. $\sqsubseteq = \subseteq_p$ lead to projections that preserve as much information as possible given $\mathcal{C}$. This is helpful to represent some inertia principle in our framework.

[2] For instance, if both $\vec{V} = \langle \{a\}, \{b, c\} \rangle$ and $\vec{V'} = \langle \{a\}, \{a, c\} \rangle$ are recoveries for B, $\vec{V'}$ should (in some contexts) be preferred because it is more uniform.

Table 1:

recovery	$\exists V_1.\phi_1$	$\exists V_2.\phi_2$	$\exists V_3.\phi_3$	$\exists V_4.\phi_4$	$\exists V_5.\phi_5$	$B \mid \vec{V}$
$\vec{V}^1 = \langle \emptyset, \{s, s_b, s_r\}, \{s, s_b, s_r\}, \{s, s_b, s_r\}, \{s, s_b, s_r\} \rangle$	ϕ_1	$c_0 \vee c_1$	$\top$	$\top$	t	$\phi_1 \wedge \neg s \wedge t$
$\vec{V}^2 = \langle \emptyset, \{t, s_b, s_r\}, \{t, s_b, s_r\}, \{t, s_b, s_r\}, \{t, s_b, s_r\} \rangle$	ϕ_1	$c_0 \vee c_1$	$\top$	s	s	$\phi_1 \wedge s \wedge \neg t$
$\vec{V}^3 = \langle \emptyset, \{c_0, c_1, c_2, s_b, s_r\}, \{c_0, c_1, c_2, s_b, s_r\}, \{c_0, c_1, c_2, s_b, s_r\}, \{c_0, c_1, c_2, s_b, s_r\} \rangle$	ϕ_1	$\top$	$s \vee t$	s	$s \wedge t$	$\phi_1 \wedge s \wedge t$
$\vec{V}^4 = \langle \emptyset, \{s_b, s_r\}, \{s_b, s_r\}, \emptyset, \{t\} \rangle$	ϕ_1	$c_0 \vee c_1$	$s \vee t$	s	s	$\phi_1 \wedge s \wedge \neg t$
$\vec{V}^5 = \langle \emptyset, \{c_0, c_1, c_2\}, \{s_b, s_r\}, \emptyset, \emptyset \rangle$	ϕ_1	$s \Rightarrow s_r$	$s \vee t$	s	$s \wedge t$	$\phi_1 \wedge s \wedge t \wedge s_r$
$\vec{V}^6 = \langle \emptyset, \{s_b, s_r\}, \emptyset, \emptyset, \{t\} \rangle$	ϕ_1	$c_0 \vee c_1$	$(s \vee t) \wedge (s \Rightarrow s_b)$	s	s	$\phi_1 \wedge s \wedge \neg t \wedge s_b$
$\vec{V}^7 = \langle \emptyset, \emptyset, \emptyset, \{s, s_b, s_r\}, \{s, s_b, s_r\} \rangle$	ϕ_1	$(c_0 \vee c_1) \wedge (s \Rightarrow s_r)$	$(s \vee t) \wedge (s \Rightarrow s_b)$	$\top$	t	$\phi_1 \wedge \neg s \wedge t$
$\vec{V}^8 = \langle \emptyset, \{c_0, c_1, c_2, s_b, s_r\}, \emptyset, \emptyset, \emptyset \rangle$	ϕ_1	$\top$	$(s \vee t) \wedge (s \Rightarrow s_b)$	s	$s \wedge t$	$\phi_1 \wedge s \wedge t \wedge s_b$
$\vec{V}^9 = \langle \emptyset, \emptyset, \{s_b, s_r\}, \emptyset, \{t\} \rangle$	ϕ_1	$(c_0 \vee c_1) \wedge (s \Rightarrow s_r)$	$s \vee t$	s	s	$\phi_1 \wedge s \wedge \neg t \wedge s_r$

Example 1 (cont'd) For instance, we have $\vec{V}^6 \subseteq_p \vec{V}^4$. Accordingly, $B \mid \vec{V}^6$ is at least as logically strong as $B \mid \vec{V}^4$.

Depending on the problem at hand, many other properties on $\sqsubseteq$ can be imposed so as to capture various intuitions about the result of the merging process.

Thus, given $B = \langle \phi_1, \ldots, \phi_n \rangle$, a preference relation $\sqsubseteq$ is said to satisfy

- the *binaricity* property if and only if $\forall \vec{V}, \vec{V}' \in \mathcal{R}_{\mathcal{C}}(B)$, if (for every $i \in 1 \ldots n$, $(V_i \neq \emptyset \Leftrightarrow V_i' \neq \emptyset)$), then $\vec{V} \sim \vec{V}'$.
- the *gathering* property if and only if $\forall \vec{V} \in \mathcal{R}_{\mathcal{C}}(B)$, $\vec{V} \sim \langle \bigcup \vec{V}, \ldots, \bigcup \vec{V} \rangle$, where $\bigcup \langle V_1, \ldots, V_n \rangle = V_1 \cup \ldots \cup V_n$ (obviously, it is required that $\langle \bigcup \vec{V}, \ldots, \bigcup \vec{V} \rangle \in \mathcal{R}_{\mathcal{C}}(B)$ for this notion to make sense).

In many cases, it is desirable to assume that $\sqsubseteq$ is a *complete* preordering. In this situation, we can equivalently represent $\sqsubseteq$ by a *ranking function* μ from $\mathcal{R}_{\mathcal{C}}(B)$ to $I\!N$ such that $\mu(\langle \emptyset, \emptyset, \ldots, \emptyset \rangle) = 0$, and $\forall \vec{V}, \vec{V}' \in \mathcal{R}_{\mathcal{C}}(B)$, if $\vec{V} \subseteq_p \vec{V}'$, then $\mu(\vec{V}) \leq \mu(\vec{V}')$. The preference relation $\sqsubseteq_\mu$ induced by μ is the total preordering defined by $\forall \vec{V} \in \mathcal{R}_{\mathcal{C}}(B)$, $\vec{V} \sqsubseteq_\mu \vec{V}'$ if and only if $\mu(\vec{V}) \leq \mu(\vec{V}')$. We say that a ranking function μ is *anonymously decomposable* if and only if there exists a total function $\mu_0 : 2^{PS} \rightarrow I\!N$ and a non-decreasing, symmetric, total function $H : I\!N^n \rightarrow I\!N$ s.t. $\mu_0(\emptyset) = 0$ and $\forall X, Y \subseteq PS$, if $X \subseteq Y$, then $\mu_0(X) \leq \mu_0(Y)$, and $H(0, \ldots, 0) = 0$, and $\mu(\vec{V}) = H(\mu_0(V_1), \ldots, \mu_0(V_n))$.

We are now in position to define the family of inference relations $\mid\!\sim^{\mathcal{C}}_{\sqsubseteq}$ that can be defined in our framework by letting $\mathcal{C}$ and $\sqsubseteq$ to vary. We call them *forget* inference relations.

Definition 6 (skeptical forget inference) *Given a belief base B and a forgetting context $\mathcal{C}$ for it, let $\sqsubseteq$ be a preference relation on $\mathcal{R}_{\mathcal{C}}(B)$ (possibly induced by a ranking function μ).*

- *A recovery $\vec{V} \in \mathcal{R}_{\mathcal{C}}(B)$ is said to be* preferred *(w.r.t. $\sqsubseteq$) if and only if it is minimal in its set w.r.t. $\sqsubseteq$, i.e., there is no $\vec{V}' \in \mathcal{R}_{\mathcal{C}}(B)$ s.t. $\vec{V}' \sqsubset \vec{V}$.*
- *Let ψ be any formula from $PROP_{PS}$. ψ is said to be (skeptically) inferred from B w.r.t. $\sqsubseteq$, denoted by $B \mid\!\sim^{\mathcal{C}}_{\sqsubseteq} \psi$, if and only if for any preferred recovery $\vec{V}$ (w.r.t. $\sqsubseteq$), we have $B \mid \vec{V} \models \psi$.*

Since skeptical inference is considered, among the preferred recoveries for B, only the maximal ones w.r.t. $\subseteq_p$ are relevant for inference, in the sense that the other ones can be ignored without modifying the inference relation $\mid\!\sim^{\mathcal{C}}_{\sqsubseteq}$. Indeed, such maximal elements correspond to the logically weakest projections.

Definition 7 (prudent forget inference) *Given*

a belief base B and a forgetting context $\mathcal{C}$ for it, let $\sqsubseteq$ be a preference relation on $\mathcal{R}_{\mathcal{C}}(B)$ (possibly induced by a ranking function μ). Let ψ be any formula from $PROP_{PS}$. ψ is a prudent consequence *of B w.r.t. $\sqsubseteq$, denoted $B \mid\!\sim^{\mathcal{C}_{prudent}}_{\sqsubseteq} \psi$, if and only if $B \mid \vec{V}^{\sqsubseteq}_{prudent} \models \psi$, where $\vec{V}^{\sqsubseteq}_{prudent}$ is a recovery for B, defined as the the pointwise union $\langle \bigcup_{\vec{V} \in min(\mathcal{R}_{\mathcal{C}}(B),\sqsubseteq)} V_1, \ldots, \bigcup_{\vec{V} \in min(\mathcal{R}_{\mathcal{C}}(B),\sqsubseteq)} V_n \rangle$ of all preferred recoveries for B w.r.t. $\sqsubseteq$ (especially, the relation does not hold whenever $\vec{V}^{\sqsubseteq}_{prudent}$ does not satisfy the constraints imposed by $\mathcal{C}$).*

Example 1 (cont'd) *If $\sqsubseteq = \sqsubseteq_p$ is chosen as a preference relation, we have $min(\mathcal{R}_{\mathcal{C}}(B), \sqsubseteq) = \{\vec{V}^5, \vec{V}^6, \vec{V}^7, \vec{V}^8, \vec{V}^9\}$. Accordingly, $\vec{V}^{\sqsubseteq}_{prudent} = \langle \emptyset, \{c_0, c_1, c_2, s_b, s_r\}, \{s_b, s_r\}, \{s, s_b, s_r\}, \{s, s_b, s_r, t\} \rangle$. We obtain $B \mid\!\sim^{\mathcal{C}}_{\sqsubseteq} \phi_1 \wedge (s \vee t)$ and $B \mid\!\sim^{\mathcal{C}_{prudent}}_{\sqsubseteq} \phi_1 \wedge (s \vee t)$.*

3.3 ON THE GENERALITY OF OUR FRAMEWORK

We now show how several well-known paraconsistent inference relations belong to the family of forget relations.

3.3.1 Reasoning from Preferred Consistent Subsets

Let us first recall the definition of inference drawn from preferred consistent subsets, also called *maxcons* inference relation (see *e.g.* [Rescher and Manor, 1970] [Fagin *et al.*, 1983] [Ginsberg, 1986] [Brewka, 1989] [Baral *et al.*, 1991] [Pinkas and Loui, 1992] [Benferhat *et al.*, 1993]). Let $B = \{\phi_1, \ldots, \phi_n\}$ be a finite set of formulas s.t. each ϕ_i $(1 \leq i \leq n)$ is consistent, and $\succeq$ a preference pre-ordering on subsets of B s.t. $\forall X, Y \subseteq B$, if $X \supseteq Y$, then $X \succeq Y$[3]. Then $B \mid\!\sim^{\forall}_{\succeq} \psi$ if and only if for any preferred (w.r.t. $\succeq$) consistent subset X of B, we have $X \models \psi$.

Proposition 1 *$\mid\!\sim^{\forall}_{\succeq}$ is a maxcons inference relation if and only if there exists a preference relation $\sqsubseteq$ on $\mathcal{R}_{\mathcal{C}_S}(B)$ satisfying binaricity and s.t. for any ψ, $\{\phi_1, \ldots, \phi_n\} \mid\!\sim^{\forall}_{\succeq} \psi$ if and only if $\langle \phi_1, \ldots, \phi_n \rangle \mid\!\sim^{\mathcal{C}_S}_{\sqsubseteq} \psi$.*

The first direction of Proposition 1 expresses that under the assumption of binaricity and standard forgetting context, inference from preferred recoveries comes down to inference from preferred maximal consistent subsets. The other direction of Proposition 1 states that the set of all maxcons inference relations is contained in the set of forget inference relations, and therefore that the latter family is at least as general as the former. In particular, if preference over recoveries is defined by $\forall \vec{V}, \vec{V}' \in \mathcal{R}_{\mathcal{C}_S}(B)$, $\vec{V} \sqsubseteq \vec{V}'$ if and only if $\{i \mid V_i \neq \emptyset\} \subseteq \{i \mid V'_i \neq \emptyset\}$ then $B \mid\!\sim^{\mathcal{C}_S}_{\sqsubseteq} \psi$ if and only if $S \models \psi$ holds for any maximal (w.r.t. set inclusion) consistent subset S of B. Other criteria such that maximum cardinality, "discrimin" or "leximin", or minimum penalty, can be recovered as well. The assumption that each ϕ_i $(1 \leq i \leq n)$ in B is consistent is necessary (and sufficient) to ensure that B is recoverable given its standard forgetting context. It can be made w.l.o.g., just because B and $B \setminus \{\phi_i \mid 1 \leq i \leq n$ and ϕ_i is inconsistent$\}$ have the same consistent subsets. As an immediate consequence of Proposition 1, our framework encompasses other important frameworks, like supernormal default theories with priorities [Brewka, 1989] and syntax-based belief revision [Nebel, 1992], as specific cases.

3.3.2 Belief Merging

A *merging operator* (see *e.g.* [Liberatore and Schaerf, 1998] [Revesz, 1997] [Konieczny and Pino Pérez, 1998] [Lin and Mendelzon, 1999]) maps any finite collection (multiset) of consistent propositional formulas $B = \{\phi_1, \ldots, \phi_n\}$, with n being unfixed, into a new consistent propositional formula $Merge(\phi_1, \ldots, \phi_n)$. When n is fixed, we define a n-merging operator $Merge_n(\phi_1, \ldots, \phi_n)$ as the restriction of $Merge$ to n-uples of formulas. In order to simplify notations, we will write $Merge(\phi_1, \ldots, \phi_n)$ instead of $Merge_n(\phi_1, \ldots, \phi_n)$ in the following.

A special class of n-merging operators is the class of *decomposable merging operators induced by (pseudo-)distances.* Let $d : \Omega \times \Omega \rightarrow I\!N$ be a total function satisfying $\forall \omega, \omega' \in \Omega$, $d(\omega, \omega') = 0$ if and only if $\omega = \omega'$, and $d(\omega, \omega') = d(\omega', \omega)$. Let $*$ be a total function from $I\!N^n$ to $I\!N$, symmetric and monotonic in each of its arguments. For every $B = \{\phi_1, \ldots, \phi_n\}$ s.t. each ϕ_i $(1 \leq i \leq n)$ is consistent, the *decomposable n-merging operator* $Merge^*_d$ induced by d and $*$ is defined by $Merge^*_d(B) = \{\omega \in \Omega \mid d(\omega, B)$ is minimal$\}$, where $d(\omega, B) = *(d(\omega, \phi_1), \ldots, d(\omega, \phi_n))$ when $B = \{\phi_1, \ldots, \phi_n\}$ and $d(\omega, \phi_i) = \min_{\omega' \models \phi_i} d(\omega, \omega')$. The n-merging inference relation $\mid\!\sim^{M}_{d,*}$ induced by d and $*$ is defined by $B \mid\!\sim^{M}_{d,*} \psi$ if and only if $Merge^*_d(\{\phi_1, \ldots, \phi_n\}) \subseteq Mod(\psi)$.

In order to characterize the decomposable n-merging inference relations as forget relations, we need to focus on the *differential* ones, *i.e.*, those based on a *differential* distance, where a distance d is said to be

[3] Among usual preference relations, we find set inclusion: $X \succeq Y$ if and only if $X \supseteq Y$, and cardinality: $X \succeq Y$ if and only if $|X| \geq |Y|$.

differential if and only if there exists a total function $f : 2^{PS} \rightarrow I\!N$ satisfying (i) $f(\emptyset) = 0$ and (ii) $\forall A,B \subseteq PS$, if $A \subseteq B$, then $f(A) \leq f(B)$, and s.t. $\forall \omega, \omega' \in \Omega$, $d(\omega,\omega') = f(Diff(\omega,\omega'))$.

Proposition 2 *$\{\phi_1,\ldots,\phi_n\} \mid\!\approx^{M}_{d,*}$ is a decomposable and differential n-merging inference relation if and only if there exists a complete preference relation $\sqsubseteq_\mu$ on $\mathcal{R}_{\mathcal{C}_S}(B)$, induced by a ranking measure μ satisfying anonymous decomposability, such that for any ψ, $\{\phi_1,\ldots,\phi_n\} \mid\!\approx^{M}_{d,*} \psi$ if and only if $\langle\phi_1,\ldots,\phi_n\rangle \mid\!\approx^{\mathcal{C}_S}_{\sqsubseteq_\mu} \psi$*

The first direction of Proposition 2 expresses that under the assumptions of completeness of $\sqsubseteq_\mu$, anonymous decomposability, and standard forgetting context, forget inference comes down to inference from a merging operator. The other direction expresses that a particularly interesting subclass of inferences from merging operators is contained in the set of forget inference relations. In particular, usual arbitration and majority merging operators (see [Konieczny and Pino Pérez, 1998]) are recovered by letting (1) $\mu_i(A) = |A|$ for each $i \in 1\ldots n$, which implies that the induced distance d is the Hamming distance between interpretations, and (2) $* = max$ or $* = +$, respectively.

3.3.3 Belief Revision

Our framework also includes some approaches to *model-based*[1] belief revision, including Satoh's $\circ_S$ [Satoh, 1988] (or equivalently, the operator given in [Delgrande and Schaub, 2000]), Dalal's $\circ_D$ [Dalal, 1988] and Weber's $\circ_W$ [Weber, 1986], provided that the belief set under consideration is not $PROP_{PS}$ (*i.e.*, it can be encoded by a consistent formula ϕ).

Given two formulas ϕ and ψ, we have:
- $Diff^{min}(\psi,\phi) = min(\{Diff(\omega,\omega') \mid \omega \in Mod(\psi), \omega' \in Mod(\phi)\},\subseteq)$;
- $||Diff^{min}(\psi,\phi)|| = min(\{|Diff(\omega,\omega')| \mid \omega \in Mod(\psi), \omega' \in Mod(\phi)\},\leq)$;
- $Mod(\phi \circ_S \psi) = \{\omega \in Mod(\psi) \mid \exists \omega' \in Mod(\phi)$ s.t. $Diff(\omega,\omega') \in Diff^{min}(\psi,\phi)\}$;
- $Mod(\phi \circ_D \psi) = \{\omega \in Mod(\psi) \mid \exists \omega' \in Mod(\phi)$ s.t. $|Diff(\omega,\omega')| = ||Diff^{min}(\psi,\phi)||\}$.
- $Mod(\phi \circ_W \psi) = \{\omega \in Mod(\psi) \mid \exists \omega' \in Mod(\phi)$ s.t. ω and ω' coincides except possibly on variables from $\bigcup Diff^{min}(\psi,\phi)\}$.

The inference relations induced by $\circ_S$, $\circ_D$ and $\circ_W$ are forget ones since:

Proposition 3 *Let $\mathcal{C}$ be the forgetting context for $\langle\psi,\phi\rangle$ s.t. $F_1 = G_1 = G_2 = H = \emptyset$ and $F_2 = PS$. Let $\sqsubseteq_S$ (resp. $\sqsubseteq_D$) be a preference relation on $\mathcal{R}_{\mathcal{C}}(B)$ s.t. $\forall \vec{V}, \vec{V'} \in \mathcal{R}_{\mathcal{C}}(B)$, $\vec{V} \sqsubseteq_S \vec{V'}$ if and only if $V_2 \subseteq V_2'$ (resp. $\vec{V} \sqsubseteq_D \vec{V'}$ if and only if $|V_2| \leq |V_2'|$). Provided that ϕ is consistent, we have:*

- *for any α, $\phi \circ_S \psi \models \alpha$ if and only if $\langle\psi,\phi\rangle \mid\!\approx^{\mathcal{C}}_{\sqsubseteq_S} \alpha$;*
- *for any α, $\phi \circ_D \psi \models \alpha$ if and only if $\langle\psi,\phi\rangle \mid\!\approx^{\mathcal{C}}_{\sqsubseteq_D} \alpha$;*
- *for any α, $\phi \circ_W \psi \models \alpha$ if and only if $\langle\psi,\phi\rangle \mid\!\approx^{\mathcal{C}_{prudent}}_{\sqsubseteq_S} \alpha$.*

3.4 And Many Others!

Up to now, we showed in this section that reasoning from preferred consistent subsets and some forms of belief merging and of belief revision are particular cases of forget inference relations. Now, there are many other forget inference relations which do not belong to any of the previous families (provided that no new propositional symbols are added).

A first example is the family of *homogeneous forget inference relations*. Such relations can be obtained by requiring the forgetting context to be homogenous, or requiring the preference relation $\sqsubseteq$ to satisfy the gathering property (which is less demanding). When homogeneous inference relations are considered, it is enough to define a preference relation on subsets of PS (instead of vectors of subsets of PS). Many such relations can be obtained by letting the preference relation on 2^{PS} vary, in quite the same manner as maxcons inference relations are obtained by letting the preference relation on subsets vary. We may for instance minimize the *set* of forgotten variables (*i.e.*, $\vec{V} \sqsubseteq \vec{V'}$ if and only if $\bigcup \vec{V} \subseteq \bigcup \vec{V'}$), or minimize the *number* of forgotten variables, or, more generally, make use of a predefined penalty function or priority preordering on variables.

Here is a variety of conclusions that can be achieved through a judicious choice of $\sqsubseteq$ on Example 1.

Example 1 (continued) *Consider the following preference relations and the induced inference relations $\mid\!\approx^{\mathcal{C}}_{\sqsubseteq}$. We define the mapping $k : 2^{Var(B)} \rightarrow I\!N$ by $k = k_c + k_s + k_t$ where $k_c(V) = 3$ if $\{c_0,c_1,c_2\} \subseteq V$, $k_c(V) = 0$ otherwise ; $k_s(V) = 3$ if $\{s,s_b,s_r\} \subseteq V$, $k_s(V) = 1$ if $V \cap \{s,s_b,s_r\} = \{s_b,s_r\}$, and $k_s(V) = 0$ otherwise ; $k_t(V) = 2$ if $t \in V$, $k_t(V) = 0$ otherwise.*

- $\sqsubseteq_1 = \sqsubseteq_p$;

[1] As evoked previously, it incorporates as well the usual syntax-based ones.

- $\sqsubseteq_2$: $\vec{V} \sqsubseteq_2 \vec{V}'$ *if and only if* $\sum_{i=1}^{5} k(V_i) \leq \sum_{i=1}^{5} k(V'_i)$
- $\sqsubseteq_3$: $\vec{V} \sqsubseteq_3 \vec{V}'$ *if and only if* $\bigcup \vec{V} \subseteq \bigcup \vec{V}'$;
- $\sqsubseteq_4$: $\vec{V} \sqsubseteq_4 \vec{V}'$ *if and only if* $k(\bigcup \vec{V}) \leq k(\bigcup \vec{V}')$;
- $\sqsubseteq_5$: $\vec{V} \sqsubseteq_5 \vec{V}'$ *if and only if* $\{i \mid V_i \neq \emptyset\} \subseteq \{i \mid V'_i \neq \emptyset\}$.

The results are synthesized in table 2. $\sqsubseteq_1$ is the pointwise inclusion relation. $\sqsubseteq_2$ is a merging inference relation. $\sqsubseteq_5$ is a maxcons inference relation. $\sqsubseteq_3$ and $\sqsubseteq_4$ correspond to homogeneous inference relations: $\sqsubseteq_3$ minimizes the set of variables forgotten everywhere while $\sqsubseteq_4$ minimizes their cumulated cost.

A second family of forget inference relations which does not degenerate into one of the previous types of inference is obtained in the situation where reliability between sources is a matter of topic[5], as explained in the following example:

Example 2 *Let be $B = \langle \phi_1, \phi_2 \rangle$, with $\phi_1 = a \wedge b$, $\phi_2 = \neg a \wedge \neg b$. Let us consider the standard forgetting context for B. Assume that source 1. is more reliable to what concerns a than to what concerns b and conversely, source 2. is more reliable to what concerns b than to what concerns a. To capture it in our framework, the following ranking function can be considered (among other possible choices): $\mu(\langle V_1, V_2 \rangle) = \mu_0(V_1) + \mu_0(V_2)$ where $\mu_0(V_i) = \sum_{v \in V_i} k_i(v)$ and $k_1(a) = 2$, $k_1(b) = 1$, $k_2(a) = 1$, $k_2(b) = 2$. Then the (unique) preferred recovery for B is $\langle \{b\}, \{a\} \rangle$ and therefore we have $B \mid\!\approx^{\mathcal{C}}_{\sqsubseteq_\mu} a \wedge \neg b$.*

Finally, in order to illustrate the flexibility offered by our framework, let us consider the two following bases:

Example 3 *Let $B = \langle \phi_1, \phi_2 \rangle$, and let $B' = \langle \phi'_1, \phi'_2 \rangle$, with:*

	$\phi_1 = (a \vee b) \wedge c$ $\phi_2 = \neg a \wedge \neg b$	$\phi'_1 = a \wedge b$ $\phi'_2 = \neg a \wedge \neg b$
$\sqsubseteq_1$	$\phi_1 \vee \phi_2$	$a \Leftrightarrow b$
$\sqsubseteq_2$	$(\neg a \vee \neg b) \wedge c$	$\top$
$\sqsubseteq_3$	$(a \vee b \vee c) \wedge (\neg a \vee b)$	$a \not\Leftrightarrow b$
$\sqsubseteq_4$	$c \wedge (\neg a \vee \neg b)$	$\top$
$\sqsubseteq_5$	$c \wedge (\neg a \vee \neg b)$	$\top$

[5]Here, a topic simply is a propositional variable x and a source i conveys some information about it as soon as ϕ_i is not independent of x, *i.e.*, $\phi_i \not\equiv \exists \{x\}.\phi_i$. For a very different approach where topics are taken into account, see [Cholvy, 1995].

What should be concluded from such inconsistent bases, especially when no information about the (absolute or relative to topics) reliability of the two sources are available? There is no absolute answer to this question! The result depends on various assumptions on the merging process, and the variety of conclusions obtained through many paraconsistent inference relations, as reported in the table, exemplifies it.

The very important point here is that our framework enables to capture each of them:

- $\sqsubseteq_1$ *corresponds to inference from maximal (w.r.t. $\subseteq$ or cardinality) consistent subsets.*
- $\sqsubseteq_2$ *(resp. $\sqsubseteq_3$) corresponds to distance-based merging where d is the Hamming distance and $* = +$ (resp. $* = \max$).*
- $\sqsubseteq_4$ *= the closure of $\sqsubseteq_p$ under gathering.*
- $\sqsubseteq_5$ *= $\sqsubseteq_p$ under the standard forgetting context.*

Compared with the weakening by inhibition mechanism at work in the coherence-based approaches, weakening by variable forgetting is typically less drastic; indeed, instead of inhibiting a whole formula – or equivalently, replacing it by $\top$ – it is possible to keep all its consequences that are not involved in any contradiction (see *e.g.* the piece of information c in Example 3). Subsequently, more information can be preserved.

4 COMPUTATIONAL ISSUES

We now investigate the computational complexity of forget inference relations. We suppose the reader familiar with computational complexity (see *e.g.* [Papadimitriou, 1994]), and especially the complexity classes NP, Δ^p_2 and Π^p_2 of the polynomial hierarchy.

We have obtained the following results:

Proposition 4

1. *Given a belief base B and a forgetting context $\mathcal{C}$ for it, determining whether B is recoverable is* NP*-complete.*
2. *Provided that the preference relation $\sqsubseteq$ on $\mathcal{R}_{\mathcal{C}}(B)$ can be tested in polynomial time, the inference problem associated to $\mid\!\approx^{\mathcal{C}}_{\sqsubseteq}$ is in Π^p_2.*
3. *If moreover $\sqsubseteq$ is induced by a ranking function μ computable in polynomial time, the inference problem associated to $\mid\!\approx^{\mathcal{C}}_{\sqsubseteq}$ is in Δ^p_2.*

These results show that the gain in generality and flexibility offered by our framework does not induce

Table 2:

preference relation	preferred recoveries that are maximal w.r.t. $\subseteq_p$	$\approx_{\sqsubseteq}^{C}$
$\sqsubseteq_1$	$\tilde{V}^5$, $\tilde{V}^6$, $\tilde{V}^7$, $\tilde{V}^8$, $\tilde{V}^9$	$\phi_1 \wedge (s \vee t)$
$\sqsubseteq_2$	$\tilde{V}^6$, $\tilde{V}^9$	$\phi_1 \wedge (s \wedge \neg t)$
$\sqsubseteq_3$	$\tilde{V}^1$, $\tilde{V}^2$, $\tilde{V}^3$	$s \vee t$
$\sqsubseteq_4$	$\tilde{V}^1$, $\tilde{V}^2$, $\tilde{V}^6$	$\phi_1 \wedge (s \oplus t)$
$\sqsubseteq_5$	$\langle \emptyset, \emptyset, \emptyset, F_4, F_5 \rangle$, $\langle \emptyset, \emptyset, F_3, \emptyset, F_5 \rangle$, $\langle \emptyset, F_2, \emptyset, \emptyset, \emptyset \rangle$	$\phi_1 \wedge (s \vee t) \wedge (s \wedge t \Rightarrow s_b) \wedge (s \wedge \neg t \Rightarrow s_r)$

a complexity shift compared with many more specific approaches (such as maxcons inferences and forms of belief revision), that are already Π_2^p-complete. This holds as well for distance-based belief merging which is Δ_2^p-complete in the general case (see [Liberatore and Schaerf, 2000] [Konieczny *et al.*, 2002]), while the problem of forget inference with a complete preference relation induced by a polytime ranking function is not harder. Hardness results for our forget inference relations (in the general case) can be easily obtained from known results for maxcons inference relations, belief revision or merging (see *e.g.* [Eiter and Gottlob, 1992] [Cayrol *et al.*, 1998] [Nebel, 1998] [Konieczny *et al.*, 2002]), thanks to the forget-based characterizations of such approaches, as reported in Section 3.3.

In the general case (especially, when $\sqsubseteq$ is not induced by a ranking function), our complexity result show that two independent sources of complexity must be dealt with. One of them lies in the possibly exponential number of preferred recoveries which must be taken into account. The other one originates from the complexity of classical entailment in propositional logic.

One way to circumvent this intractability consists in *compiling* the belief base [Cadoli and Donini, 1997] [Coste-Marquis and Marquis, 2000]. Such a compilation may consist in computing off-line the set $min(\mathcal{R}_C(B), \sqsubseteq)$ - or only the maximal elements of it w.r.t. $\subseteq_p$. In this situation, the source of complexity due to the number of preferred recoveries disappears (in practice, this is significant only when the number is small enough, which is always the case when prudent inference is considered); accordingly, the complexity of (skeptical or prudent) inference goes down to coNP. The other source of complexity can also be removed by imposing some restrictions on the belief base B. In this situation, on-line forget inference is tractable.

Proposition 5

1. *Provided that $min(\mathcal{R}_C(B), \sqsubseteq)$ - or only the maximal elements of it w.r.t. $\subseteq_p$ - is part of the input, the inference problem associated to $\approx_{\sqsubseteq}^{C}$ (or $\approx_{\sqsubseteq}^{C_{prudent}}$) is in* coNP.
2. *If moreover the set $\{\phi_i \mid 1 \leq i \leq n\}$ of formulas from B belongs to a class of formulas that are tractable for clausal entailment and stable for new variable renaming[6], the inference problem associated to $\approx_{\sqsubseteq}^{C}$ (or $\approx_{\sqsubseteq}^{C_{prudent}}$) is in* P *when CNF queries are considered.*

Interestingly, several well-known classes that are tractable for the satisfiability problem are also stable for new variable renaming; For example, this is the case for the class of Krom formulas (CNF formulas consisting of binary clauses) and for the class of CNF Horn renamable formulas.

5 CONCLUSION

We have proposed a new framework for reasoning from inconsistent belief bases, which proceeds by selecting subsets of variables to be forgotten in the pieces of information, respecting some fixed constraints. We have shown that this framework encompasses many existing frameworks as specific cases, namely reasoning from preferred consistent subsets, forms of belief merging and of belief revision.

As evoked in the introduction, our approach is based on classical entailment so that the logical contents of the belief base is weakened instead of the inference relation itself. This gives to our family of forget relations some pros and some cons. Thus, if the belief base consists of a single inconsistent formula, our approach is not adapted (as well as the coherence-based approaches and belief merging, which are also based on classical entailment). Paraconsistent logics based on subclassical entailment relations can then be used with profit. Conversely, since weakening bears on the

[6] This means that for every formula ϕ from $PROP_{PS}$ and for every substitution σ that renames some occurrences of variables from ϕ into new variables (not occurring in ϕ), if ϕ belongs to the class under consideration, this is also the case for $\sigma(\phi)$.

inference relation in paraconsistent logics, it is uniform and the fact that information comes from several sources is not exploited. For instance, it is not possible to derive b from $\{a \wedge (\neg a \vee b), \neg a\}$ using the paraconsistent logic LP_m [Priest, 1991] (this is because $\neg a \vee b$ is a consequence of $\neg a$).

The idea of weakening beliefs instead of throwing them out is not new. For instance, in [Bessant *et al.*, 2001], each formula belonging to a conflict is weakened through the introduction of a new propositional symbol. This way, inconsistency is removed. Then, non-monotonic inference is conducted over the resulting base so as to prefer the introduced symbols as false. This enables a non-trivial treatment of iterated revision. Unfortunately, this approach comes down to the well-known WIDTIO approach ("When In Doubt Throw It Out") [Winslett, 1990] when only one revision is performed. Indeed, in this situation, weakening by variable introduction does not enable more information to be preserved than if beliefs were removed. Our approach does not suffer from this drawback because weakening is achieved through forgetting instead of variable introduction. This also enables classical entailment to be performed from the resulting base (i.e., after forgetting) instead of nonmonotonic inference.

Finally, weakening formulas by variable forgetting can be useful in many practical situations. For instance, if a series of tests has to be performed so as to "solve" inconsistency [Lang and Marquis, 2000; Liberatore and Donini, 2000], preferred recoveries give some hints about the variables that should be tested first, namely, the most important ones (w.r.t. the graph G in $\mathcal{C}$) that have to be forgotten. Similarly, in a decision-theoretic context where the ϕ_i's are no longer *beliefs* but *preferences* of several agents, variables that do not need to be forgotten are these the agents agree on, and preferred recoveries help finding the variables about which negociation should start.

Acknowledgements

The authors wish to thanks Philippe Besnard and Torsten Schaub for many helpful discussions. The second author has been partly supported by the IUT de Lens, the Université d'Artois, the Région Nord/Pas-de-Calais under the TACT-TIC project, and by the European Community FEDER Program.

References

[Baral *et al.*, 1991] C. Baral, S. Kraus, and J. Minker. Combining multiple knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 3(2):208–220, 1991.

[Benferhat *et al.*, 1993] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, and H. Prade. Inconsistency management and prioritized syntax- based entailment. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI'93)*, pages 640–645, 1993.

[Besnard and Hunter, 1998] Ph. Besnard and A. Hunter. *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 2, chapter Introduction to actual and potential contradictions, pages 1–11. Kluwer Academic Publishers, 1998.

[Bessant *et al.*, 2001] B. Bessant, E. Grégoire, P. Marquis, and L. Saïs. *Frontiers of Belief Revision*, volume 22 of *Applied Logic Series*, chapter Iterated syntax-based revision in a nonmonotonic setting, pages 369–391. Kluwer Academic, 2001.

[Brewka, 1989] G. Brewka. Preferred subtheories: an extended logical framework for default reasoning. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI'89)*, pages 1043–1048, Detroit (MI), 1989.

[Cadoli and Donini, 1997] M. Cadoli and F.M. Donini. A survey on knowledge compilation. *AI Communications*, 10:137–150, 1997. Printed in 1998.

[Cayrol *et al.*, 1998] C. Cayrol, M.-C. Lagasquie-Schiex, and Th. Schiex. Nonmonotonic reasoning: from complexity to algorithms. *Annals of Mathematics and Artificial Intelligence*, 22(3–4):207–236, 1998.

[Cholvy, 1995] L. Cholvy. Automated reasoning with merged contradictory information whose reliability depends on topics. In *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU'95)*, pages 125–132, Fribourg, 1995.

[Coste-Marquis and Marquis, 2000] S. Coste-Marquis and P. Marquis. Compiling stratified belief bases. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI'00)*, pages 23–27, Berlin, 2000.

[Dalal, 1988] M. Dalal. Investigations into a theory a knowledge base revision: preliminary report. In *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI'88)*, pages 475–479, St Paul (MN), 1988.

[Delgrande and Schaub, 2000] J. Delgrande and T. Schaub. A consistency-based model for belief change: Preliminary report. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI'98)*, pages 392–398, 2000.

[Eiter and Gottlob, 1992] Th. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactuals. *Artificial Intelligence*, 57:227–270, 1992.

[Fagin *et al.*, 1983] R. Fagin, J.D. Ullman, and M.Y. Vardi. On the semantics of updates in databases. In *Proceedings of the 2nd ACM Syposium on Principles of Database Systems (PODS'83)*, pages 352–355, 1983.

[Ginsberg, 1986] M.L. Ginsberg. Counterfactuals. *Artificial Intelligence*, 30:35–79, 1986.

[Hunter, 1998] A. Hunter. *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 2, chapter Paraconsistent logics, pages 11–36. Kluwer Academic Publishers, 1998.

[Konieczny and Pino Pérez, 1998] S. Konieczny and R. Pino Pérez. On the logic of merging. In *Proceedings of the 6th International Conference on Knowledge Representation and Reasoning (KR'98)*, pages 488–498, Trento, 1998.

[Konieczny *et al.*, 2002] S. Konieczny, J. Lang, and P. Marquis. Distance-based merging: a general framework and some complexity results. In *Proceedings of the 8th International Conference on Knowledge Representation and Reasoning (KR'02)*, Toulouse, 2002. This issue.

[Lang and Marquis, 1998] J. Lang and P. Marquis. Complexity results for independence and definability. In *Proceedings of the 6th International Conference on Knowledge Representation and Reasoning (KR'98)*, pages 356–367, Trento, 1998.

[Lang and Marquis, 2000] J. Lang and P. Marquis. In search of the right extension. In *Proceedings of the 7th International Conference on Knowledge Representation and Reasoning (KR'00)*, pages 625–636, Breckenridge (CO), 2000.

[Lang *et al.*, 2001] J. Lang, P. Liberatore, and P. Marquis. The complexity of propositional independence. Technical report, IRIT, 2001. (submitted for publication).

[Liberatore and Donini, 2000] P. Liberatore and F.M. Donini. Verification programs for abduction. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI'00)*, pages 166–170, Berlin, 2000.

[Liberatore and Schaerf, 1998] P. Liberatore and M. Schaerf. Arbitration (or how to merge knowledge bases). *IEEE Transactions on Knowledge and Data Engineering*, 10(1):76–90, 1998.

[Liberatore and Schaerf, 2000] P. Liberatore and M. Schaerf. Brels: A system for the integration of knowledge bases. In *Proceedings of the 7th International Conference on Knowledge Representation and Reasoning (KR'00)*, pages 145–152, Breckenridge (CO), 2000.

[Lin and Mendelzon, 1999] J. Lin and A.O. Mendelzon. Knowledge base merging by majority. In *Dynamic Worlds: From the Frame Problem to Knowledge Management*. Kluwer, 1999.

[Lin and Reiter, 1994] F. Lin and R. Reiter. Forget it! In *Proceedings of the AAAI Fall Symposium on Relevance*, pages 154–159, New Orleans (LA), 1994.

[Nebel, 1992] B. Nebel. *Belief Revision*, chapter Syntax-based approaches to belief revision, pages 52–88. Number 29 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1992.

[Nebel, 1998] B. Nebel. *Belief Revision*, volume 3 of *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, chapter How hard is it to revise a belief base?, pages 77–145. Kluwer Academic, 1998.

[Papadimitriou, 1994] Ch. H. Papadimitriou. *Computational complexity*. Addison–Wesley, 1994.

[Pinkas and Loui, 1992] G. Pinkas and R.P. Loui. Reasoning from inconsistency: a taxonomy of principles for resolving conflict. In *Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning (KR'92)*, pages 709–719, Cambridge (MA), 1992.

[Priest, 1991] G. Priest. Minimally inconsistent LP. *Studia Logica*, 50:321–331, 1991.

[Rescher and Manor, 1970] N. Rescher and R. Manor. On inference from inconsistent premises. *Theory and Decision*, 1:179–219, 1970.

[Revesz, 1997] P. Z. Revesz. On the semantics of arbitration. *International Journal of Algebra and Computation*, 7(2):133–160, 1997.

[Satoh, 1988] K. Satoh. Nonmonotonic reasoning by minimal belief revisions. In *Proceedings of the International Conference on Fifth Generation Computer Systems (FGCS'88)*, pages 455–462, Tokyo, 1988.

[Weber, 1986] A. Weber. Updating propositional formulas. In *Proceedings of the 1st Conference on Expert Database Systems*, pages 487–500, 1986.

[Winslett, 1990] M. Winslett. *Updating logical databases*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.

Appendix: proof sketches

Sketch of proof of Proposition 1:
($\Leftarrow$): Let $\sqsubseteq$ be a preference relation over $\mathcal{R}_{\mathcal{C}_S}(B)$ satisfying binaricity. Let $B = \langle \phi_1, \ldots, \phi_n \rangle$. For any subset S of $\{\phi_1, ..., \phi_n\}$, we define $\vec{V}_S = \langle V_1, \ldots, V_n \rangle$ where for each $i \in 1 \ldots n$, $V_i = \emptyset$ if $\phi_i \in S$ and $V_i = F_i$ otherwise. Now, we define a preference relation $\succeq$ on consistent subsets of $\{\phi_1, ..., \phi_n\}$ by $X \succeq Y$ if and only if $\vec{V}_X \sqsubseteq \vec{V}_Y$. It can then be shown that $\{\phi_1, ..., \phi_n\} \mid\!\sim^{\forall}_{\succeq} \psi$ if and only if $B \mid\!\sim^{\mathcal{C}_S}_{\sqsubseteq} \psi$.
($\Rightarrow$): Let $\succeq$ be a preference relation over consistent subsets of $\{\phi_1, ..., \phi_n\}$. For any vector $\vec{V} = \langle V_1, \ldots, V_n \rangle$ of subsets of PS, let $S_{\vec{V}}$ be the subset of $\{\phi_1, ..., \phi_n\}$ defined by $S_{\vec{V}} = \{\phi_i \mid V_i = \emptyset\}$. Let us now define the preference relation $\sqsubseteq$ over $\mathcal{R}_{\mathcal{C}_S}(B)$ by $\vec{V} \sqsubseteq \vec{V'}$ if and only if $S_{\vec{V}} \succeq S_{\vec{V'}}$. Note that $\sqsubseteq$ satisfies binaricity. Now, it can be shown that $\{\phi_1, ..., \phi_n\} \mid\!\sim^{\forall}_{\succeq} \psi$ if and only if $\langle \phi_1, ..., \phi_n \rangle \mid\!\sim^{\mathcal{C}_S}_{\sqsubseteq} \psi$.

Sketch of proof of Proposition 2:
($\Leftarrow$): Assume that $\sqsubseteq_\mu$ is a complete preference relation on $\mathcal{R}_{\mathcal{C}_S}(B)$ induced by a ranking function μ satisfying decomposability, *i.e.*, there exists μ_0 and H s.t. $\mu(\vec{V}) = H(\mu_0(V_1), \ldots, \mu_0(V_n))$. Let us define the merging operator $\Delta^{M}_{d,*}$ induced from the pseudo-distance defined by $d(\omega, \omega') = \mu_0(Diff(\omega, \omega'))$ and the aggregation function $* = H$. It can then be proven that $B \mid\!\sim^{M}_{d,*} \psi$ if and only if $B \mid\!\sim^{\mathcal{C}_S}_{\sqsubseteq_\mu} \psi$.
($\Rightarrow$) Let $B \mid\!\sim^{M}_{d,*}$ be a decomposable and differential n-merging inference relation induced by pseudo-distance d and aggregation function $*$, *i.e.*, $d(\omega, B) = *_{i=1}^{n}(d(\omega, \phi_i))$ and $d(\omega, \omega') = f(Diff(\omega, \omega'))$. Let us consider the forget inference induced from the standard forgetting context and the function μ defined by $\mu(\vec{V}) = H(\mu_0(V_1), \ldots, \mu_0(V_n))$ where $\mu_0 = f$ and $H = *$. Then we prove that we get $\mu(\omega) = d(\omega, B)$, from which it follows that $B \mid\!\sim^{M}_{d,*} = B \mid\!\sim^{\mathcal{C}_S}_{\sqsubseteq_\mu}$.

Sketch of proof of Proposition 3:
We give the proof in the case of $\circ_S$. As a direct consequence of Corollary 3 from [Lang *et al.*, 2001], for every $\omega \in Mod(\psi)$, there exists $\omega' \in Mod(\phi)$ s.t. $Diff(\omega, \omega') = S$ if and only if $\omega \models \psi \wedge \exists S.\phi$ holds. Subsequently, we have $\{\omega \in Mod(\psi) \mid \exists \omega' \in Mod(\phi)$ s.t. $Diff(\omega, \omega') \in Diff^{min}(\psi, \phi)\} = \{\omega \in Mod(\psi \wedge \exists S.\phi) \mid S \in Diff^{min}(\psi, \phi)\}$. Now, $S \in Diff^{min}(\psi, \phi)$ if and only if S is a minimal (w.r.t. $\subseteq$) set of variables s.t. $\psi \wedge \exists S.\phi$ is consistent. This is equivalent to state that $\langle \emptyset, S \rangle$ is a preferred recovery for $\langle \psi, \phi \rangle$ w.r.t. $\sqsubseteq_S$ given $\mathcal{C}$, and the conclusion follows. The proof for the case $\circ_D$ is similar, with cardinality playing the role that set inclusion plays for $\circ_S$. We omit the case of $\circ_W$, the proof of which is not hard to establish.

Sketch of proof of Proposition 4: The membership proof for Point 1 goes as follows. Let $B = \langle \phi_1, \ldots, \phi_n \rangle$ be a belief base and let $\mathcal{C} = \langle F, G, H \rangle$ be a forgetting context for B. The following nondeterministic algorithm can be used to determine whether B is recoverable: 1. Guess n subsets $V_1, \ldots, V_n$ of $Var(B)$ and $n+1$ interpretations $\omega, \omega_1, \ldots, \omega_n$ over $Var(B)$; 2. For each $i \in 1 \ldots n$, check that: (a) $V_i \subseteq F_i$; (b) for every $(v, v') \in G_i$, if $v \in V_i$, then $v' \in V_i$; (c) $\omega_i \models \phi_i$; (d) ω coincides with ω_i on $Var(\phi_i) \setminus V_i$.
Due to lack of space we omit hardness (easy to establish) and points 2 and 3.

Sketch of proof of Proposition 5: Let $\vec{V} = \langle V_1, \ldots, V_n \rangle$ be any preferred recovery from $min(\mathcal{R}_{\mathcal{C}}(B), \sqsubseteq)$. Forget inference from B comes down to classical inference from $\bigvee_{\vec{V} \in min(\mathcal{R}_{\mathcal{C}}(B), \sqsubseteq)} B \mid \vec{V}$. The key point is the equivalence stating that $\{\exists V_1.\phi_1, \ldots, \exists V_n.\phi_n\} \models \psi$ if and only if $\bigwedge_{i=1}^{n} \phi_i[new(V_i)] \models \psi$, where $\phi_i[new(V_i)]$ is the formula obtained by renaming in an uniform way in ϕ_i every variable $v \in V_i$ by the new variable v^i, provided that ψ does not contain any new variable v^i, $i \in 1 \ldots n$. While the formula corresponding to $B \mid \vec{V}$ after such a renaming process is not logically equivalent to it[7], it is *query-equivalent* to it (*i.e.*, equivalence in preserved over the original language, only), but this is sufficient for our query answering purpose.

[7] In [Lang *et al.*, 2001], we show that in the general case and unless NP $\cap$ coNP $\subseteq$ P/poly, there exists no propositional formula α equivalent to $\exists V.\phi$, s.t. the size of α is polynomially bounded in the size of ϕ plus the size of V.

Updating Contexts

Luciano Serafini
ITC – IRST
38050 Povo, Trento, Italy
serafini@itc.it

Antonia Donà
ITC – IRST
38050 Povo, Trento, Italy
antodona@itc.it

Abstract

Different formalisms for contexts have been successfully used to represent and reason about distributed knowledge. In these formalisms, knowledge is represented as a set of contexts, each representing a piece of the whole knowledge. Contexts have been proved particularly adapt to deal with heterogeneous distributed knowledge, but, in order to deal with the change of such a knowledge, they must be extended. In particular, they must cope with the problems of adding, deleting, or changing facts in a context, and computing the effect of this change in the other contexts. The current approaches to belief revision and multi-agent belief revision can be helpful, but they do not provide a satisfactory treatment of heterogeneity. We provide a formal definition of the operation of updating a context, and we define an algorithm that computes the effects of updating a context on other related contexts. We take a semantic perspective, i.e., each context is formalized as a set of possible partial models of the world.

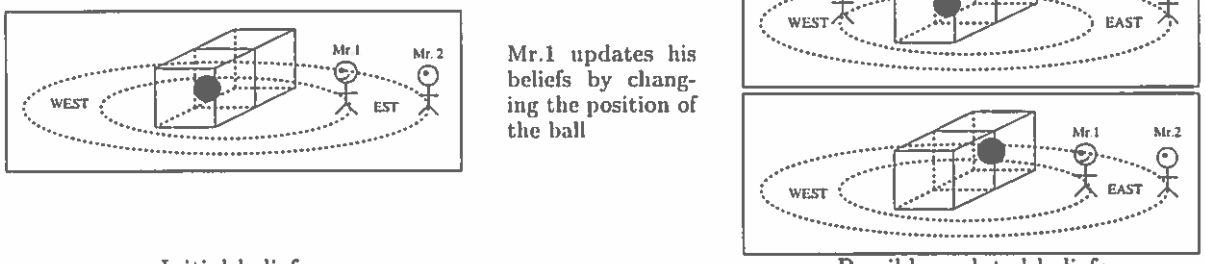
1 INTRODUCTION

In a distributed environment, such as the information system of a large company, or the beliefs of a set of agents, or, to the extreme, the whole world wide web, knowledge is not represented as a monolithic system. Rather it is scattered in a number of relatively "small" and specialized pieces, called local knowledge bases. Each local knowledge base adopts a language to represent its knowledge. This language can either be shared among all, or a subset of the local knowledge bases, or it could be a specialized language. Furthermore, each local knowledge base gives a way to interpret (i.e. it associates with a semantics) the statements of its language in terms of real world entities (i.e. propositions or objects). In distributed knowledge we cannot assume that this semantics is the same for all the local knowledge bases, but the well known phenomenon (and problem) of semantic heterogeneity arises. Semantic heterogeneity occurs when the same or related real world entities can be represented, at a different level of abstraction and by different statements, in more than one local knowledge base. This means that local knowledge bases are not independent, rather they overlap.

Due to overlapping, changing a piece of information in a knowledge base results both in an update of the local knowledge base and in a revision of the knowledge in those bases which overlap with the original one. These revisions might in turn lead to the revising of other knowledge bases, and so on. Determining the minimal update propagation in order to maintain consistency in a distributed knowledge base has recently become a very relevant problem. The main approaches in this area are those in [van der Meyden, 1994], [Kfir-Dahav and Tennenholz, 1996], [Malheiro and Oliveira, 2000], [Liu and Williams, 1999] and [Liu and Williams, 2000]. These approaches, however, do not provide a satisfactory solution for revising distributed knowledge (or beliefs) in the presence of heterogeneity. The main reason is that all rely on the fact that shared knowledge, (i.e. the overlaps between local knowledge bases), has a uniform representation. This assumption, however, in many cases, is not realistic. Consider the following example.

Example 1. Consider the scenario in which $Mr.1$ and $Mr.2$ are two agents looking at a box composed of two slots, each possibly containing a ball (see Figure 1).

Suppose that $Mr.1$ and $Mr.2$ can move around the box along two circular tracks, and that $Mr.1$ moves on the internal track, while $Mr.2$ moves on the external



Initial beliefs

Possible updated beliefs

Figure 1: *An example.*

one. According to what they can see, $Mr.1$'s beliefs regard balls being in the right or left slot of the box, and $Mr.2$'s beliefs regard balls being in the left/right slot, and a man being between himself and the box. For instance, in the situation shown on the left side of Figure 1, $Mr.1$ believes that there is a ball on the left slot, and that there is no ball on the right slot (in symbols $L \wedge \neg R$). $Mr.2$ believes that there is a ball on the left slot, that there is no ball on the right slot, and that there is a man in front of him (in symbol $L \wedge \neg R \wedge M$).

Notice that in this example, the relation between the meaning of L in $Mr.1$ and $Mr.2$ beliefs depends on the value of M in $Mr.2$ beliefs. If M is true, then L has the same meaning in both $Mr.1$ and $Mr.2$'s beliefs. If M is false, the meaning of L in $Mr.1$'s beliefs is independent from the meaning of L in $Mr.2$'s beliefs as they are referring to two different slots.

Suppose that, after a while $Mr.1$ is looking at the box and he sees that the ball has been moved from the left side to the right side of the box. He therefore updates his beliefs to $\neg L \wedge R$. Suppose also that $Mr.1$ communicates to $Mr.2$ his new beliefs. How does $Mr.2$ react to this, if he trusts $Mr.1$? He has two options, either to believe that $Mr.1$ has changed his position going to other side of the box, or that the ball has been moved from the left to the right. In the first case $Mr.1$ updates its beliefs to $L \wedge \neg R \wedge \neg M$, in the second case his beliefs are updated to $\neg L \wedge R \wedge M$. These two updates correspond to the two scenes shown on the right of Figure 1.

A formalization of distributed belief revision should provide a solution to the following problem: consider the implementation of two autonomous computer programs (agents) that simulate $Mr.1$ and $Mr.2$ reasoning, without having access to a description of the state of the world (in terms of $Mr.1$ and $Mr.2$ positions and the positions of the balls in the box), but simply relying on the coordination between the two agents. If, for any reason one agent decides to revise its beliefs, how does the other agent revise its own beliefs to maintain them consistent with a possible (not accessible) state of the world?

In this paper, we propose a new logic based approach to the revision of distributed knowledge assuming there is no global representation of the knowledge, and that information in different knowledge bases can be related (e.g., the same information is represented in two or more knowledge bases, the information in a knowledge base is constrained to the information in a different knowledge base).

Our approach is based on the theory of context. Contexts formalisms have been proposed in AI by [Buvač et al., 1995], [Buvač and Mason, 1993], and by [Giunchiglia and Serafini, 1994], [Ghidini and Giunchiglia, 2001] and [Serafini and Giunchiglia, 2000]. Furthermore, in [Serafini and Ghidini, 2000] and [McCarthy and Buvač, 1998] *contexts* have been proposed for the representation of distributed knowledge in the presence of heterogeneity. The theory so far, however, does not provide any support for revising the contents of contexts. In this paper, we give a formal definition of contextual update, and we provide a sound and complete procedure for computing the possible outcomes of a contextual update.

The structure of the paper is the following. In Section 2, we introduce the formalisms, based on contexts, which we have adopted. In Section 3, we give a set of criteria on the update of context, which lead us to an implicit definition of the update operation. In Section 4, we provide an algorithm that computes this update operator. We conclude the paper by a comparison with similar approaches (Section 5).

2 LOGIC OF CONTEXTS

To represent distributed knowledge we adopt the logic of contexts developed in [Giunchiglia and Serafini, 1994] and [Ghidini and Giunchiglia, 2001]. In this section we recall the key concepts of this logic.

Logic of context represents knowledge in a set of theories (contexts) each of them constituting a partial, approximate, and perspectival description of the world [Giunchiglia, 1993], [Benerecetti et al., 2000]. Each context partially describes a portion of the world from a perspective. For instance, the context of $Mr.1$'s beliefs partially describes the content of the box, as $Mr.1$ might not believe where a ball is. It describes only a portion of the world, as in $Mr.1$'s context there are no beliefs about the position of $Mr.2$; finally, it describes the world from $Mr.1$'s contextual perspective, because the meaning of the formulas in this context depends on $Mr.1$'s position.

To each context we associate a logical language that, for the purpose of this paper, is supposed to be a propositional language. Formally let $\{L_i\}_{i\in I}$ be a family of propositional languages defined over a set of indices I (in the following simply $\{L_i\}$). Intuitively, I contains the names of the contexts in which knowledge (beliefs) is partitioned and each L_i is the content language of the i-th context.

In our example, we can define the propositional languages L_1 and L_2 used by $Mr.1$ and $Mr.2$ to describe their views. L_1 is the propositional language with the primitive propositions $\{L, R\}$ and L_2 is the propositional language with the primitive propositions $\{L, R, M\}$.

The meaning of formulas depends on the context where they are stated. For instance, the meaning of L in context 1 is that $Mr.1$ believes that there is a ball on the left side of the box. But the left depends on the position of $Mr.1$. The proposition M in the context 1 has no meaning at all. To distinguish the different meaning of propositions, let us write $i:\phi$ to mean ϕ and that ϕ is a formula of L_i. We say that ϕ is an L_i-formula, and that $i:\phi$ is a formula.

The semantics of a family of languages $\{L_i\}$, called Local Model Semantics, is based on the notion of chain. Chains are defined in terms of propositional models of the languages $\{L_i\}_{i\in I}$. A propositional model m of L_i is a consistent and complete set of literals in L_i, i.e., a set m such that, for any primitive proposition p of L_i, either $p \in m$ or $\neg p \in m$, but not $\{p, \neg p\} \subseteq m$.

Definition 1 (Chain). A *chain* is a function c that associates to each index $i \in I$ a set of models of L_i, such that there is at least an i with $c(i) \neq \emptyset$. We write c_i instead of $c(i)$. A chain c satisfies a formula $i:\phi$, in symbols $c \models i:\phi$, if for any $m \in c_i$, $m \models \phi$, according to the definition of satisfiability ($\models$) in propositional logic.

Intuitively, a chain represents a combination of epistemic states, one for each context. The epistemic state associated to each context is a set of possible states of the "piece of world" described from the perspective of the context. Two chains in the languages for Example 1 are:

$$c = \langle \boxed{\bullet\ \ }, \boxed{\bullet\ \ \mathrm{M}} \rangle \quad (1)$$

$$c' = \langle \boxed{\bullet\ \bullet}, \{\boxed{\bullet\ \ \mathrm{M}}, \boxed{\ \ \bullet\ \mathrm{M}}\} \rangle \quad (2)$$

The graphical notation $\boxed{\bullet\ \ }$ represents the model $\{L, \neg R\}$ of the language L_1. Similarly, $\boxed{\bullet\ \ \mathrm{M}}$ represents the model $\{L, \neg R, M\}$ of L_2. The other graphical notations have analogous meaning. Furthermore, if a set is composed of a single element we omit the curly brackets.

The chain c represents the (possible) situation where $Mr.1$ and $Mr.2$ both believe that there is ball on the left side and no ball on the right side and $Mr.2$ believes also that there is a man between himself and the box. This combination of epistemic states is possible as it corresponds to the situation shown on the left box of Figure 1. Notice, however, that c is also consistent with the situation where $Mr.1$ and $Mr.2$ are both on the west side of the box, and the ball is moved to the other slot. The local perspectives of $Mr.1$ and $Mr.2$, however, cannot discriminate between these two situations. c' represents a situation where $Mr.1$ believes that there are two balls in the box, while $Mr.2$ believes that there is only one ball. This is an *impossible* combination of epistemic states, as there is no real situation where this can happen.

Definition 2 (Compatibility relation). A *compatibility relation* is a non empty set C of chains. A compatibility relation C satisfies $i:\phi$, in symbols $C \models i:\phi$, if, for each chain $c \in C$, $c \models i:\phi$.

Intuitively, a compatibility relation contains the combinations of epistemic states compatible with a "real situation". For instance, the compatibility relation associated with Example 1 contains the chain c and does not contain chain c' defined in (1) and (2) respectively. c is indeed compatible with the situation shown on the left box of Figure 1 while c' is not compatible with any possible state of the world of this example. The fact that a chain actually corresponds to a real situation, however, is not explicitly represented in the semantics, as a compatibility relation does not contain any

formalization of the state of the world.

Despite the fact that a compatibility relation can be *any* set of chains, practical use, restricts us to consider compatibility relations that can be finitely specified using some specification language. From the theory of context defined in [Giunchiglia and Serafini, 1994] we take the notion of *bridge rules*. Bridge rules allow to relate the truth of a formula in a context i with the truth of a set of formulas in other contexts.

Definition 3 (Bridge rule). Let $i_1, \ldots, i_n, j \in I$, $i_1{:}\phi_1, \ldots, i_n{:}\phi_n$ and $j{:}\psi$ be propositional formulas. A *bridge rule* is an expression of the form: $i_1{:}\phi_1, \ldots, i_n{:}\phi_n \rightarrow j{:}\psi$. A chain c satisfies a bridge rule in case, if $c \models i_1{:}\phi_1, \ldots, c \models i_n{:}\phi_n$, then $c \models j{:}\psi$.

A set of bridge rules BR defines the compatibility relation C_{BR} composed of all the chains that satisfy BR. Logical consequence in these compatibility relations captures the relations among formulas in different contexts, modeled by a compatibility relation.

Definition 4 (Logical Consequence w.r.t. a set of BR). A formula $i : \phi$ is a *logical consequence* of a set of formulas Γ w.r.t. a set of bridge rules BR ($\Gamma \models_{BR} i{:}\phi$) if, for all $c \in C_{BR}$, if $c \models \Gamma$, then $c \models i{:}\phi$

Example 2. A set of bridge rules that formalize the constraints between the beliefs of $Mr.1$. and $Mr.2$ for Example 1 are shown in Figure 2.

Bridge rules have a declarative meaning, which is the one provided by the definition of satisfiability (Definition 3). They also have a procedural meaning in terms of update propagation. For instance, the bridge rules BR1, BR2, BR5 and BR6 state that updates on L in 1 and M in 2 affect the update of L and R in 2. Similar interpretations hold for all the other bridge rules. Notice that, with this intuitive interpretation, M cannot be changed as an effect of an update on L and R, as M does not appear in the consequence of any bridge rule.

3 THE CONTEXTUAL UPDATE OPERATOR

Example 3. Consider once again Example 1. Suppose that both $Mr.1$ and $Mr.2$ believe that there is a ball on the left side and no balls on the right side of the box, and that $Mr.2$ believes that $Mr.1$ is in front of him. The chain representing the beliefs of $Mr.1$ and $Mr.2$ is the following:

$$c = \langle c_1 = \boxed{\bullet\,|\,\,}, \; c_2 = \boxed{\bullet\,|\,\,|\,\mathrm{M}} \rangle \qquad (3)$$

Now suppose that $Mr.1$ updates his beliefs based on the fact that he perceives that the ball moves from the left to the right side of the box. The chain formalizing this situation is:

$$c' = \langle c'_1 = \boxed{\,\,|\,\bullet}, \; c'_2 = \boxed{\bullet\,|\,\,|\,\mathrm{M}} \rangle \qquad (4)$$

This chain does not satisfy bridge rules BR3, BR5, BR9, and BR15. Indeed it describes an impossible situation. We need to minimally update $Mr.2$, in order to obtain a chain that satisfies all the bridge rules. There are the following two possibilities:

$$c'' = \langle c''_1 = \boxed{\,\,|\,\bullet}, \; c''_2 = \boxed{\bullet\,|\,\,|\,\,} \rangle \qquad (5)$$

$$c''' = \langle c'''_1 = \boxed{\,\,|\,\bullet}, \; c'''_2 = \boxed{\,\,|\,\bullet\,|\,\mathrm{M}} \rangle \qquad (6)$$

c'', however, is not acceptable as it involves a change of M, without $Mr.2$ having decided to change explicitly. We therefore select c'''. Finally, suppose that $Mr.2$ decides to update M obtaining the following chain:

$$c'''' = \langle c''''_1 = \boxed{\,\,|\,\bullet}, \; c''''_2 = \boxed{\,\,|\,\bullet\,|\,\,} \rangle \qquad (7)$$

Again c'''' does not satisfies the bridge rules. In this case, however, there is not a unique possibility. Possible changes to restores consistency are the following:

$$c^1 = \langle c^1_1 = \boxed{\,\,|\,\bullet}, \; c^1_2 = \boxed{\bullet\,|\,\,|\,\,} \rangle \qquad (8)$$

$$c^2 = \langle c^2_1 = \boxed{\bullet\,|\,\,}, \; c^2_2 = \boxed{\,\,|\,\bullet\,|\,\,} \rangle \qquad (9)$$

$$c^3 = \langle c^3_1 = \boxed{\bullet\,|\,\bullet}, \; c^3_2 = \boxed{\bullet\,|\,\bullet\,|\,\,} \rangle \qquad (10)$$

$$c^4 = \langle c^4_1 = \boxed{\,\,|\,\,}, \; c^4_2 = \boxed{\,\,|\,\,|\,\,} \rangle \qquad (11)$$

In terms of number of changes the above adjustments involve two changes each and therefore there is no theoretical a priori reason to prefer one to the other. By taking a skeptical approach, we leave open the possibility of all of them, by adopting a compatibility relation, instead of a single chain. Therefore we have that the new beliefs of $Mr.1$ and $Mr.2$ are formalized by the compatibility relation composed of the four chains.

Let us now give a formal definition of update so that the previous example can be represented in a systematic way.

3.1 IMPLICIT DEFINITION

A contextual update operator takes a local update (an update in a context) a compatibility relation and returns a compatibility relation in which the local update is performed and all the bridge rules are satisfied. Local updates are specified in a *local update language*.

Definition 5 (Local update languages). Given a propositional language L a *local update* in L is defined as follows:

(BR1)	$1{:}L, 2{:}M$	$\rightarrow$	$2{:}L$	(BR2)	$1{:}L, 2{:}\neg M$	$\rightarrow$	$2{:}R$
(BR3)	$1{:}R, 2{:}M$	$\rightarrow$	$2{:}R$	(BR4)	$1{:}R, 2{:}\neg M$	$\rightarrow$	$2{:}L$
(BR5)	$1{:}\neg L, 2{:}M$	$\rightarrow$	$2{:}\neg L$	(BR6)	$1{:}\neg L, 2{:}\neg M$	$\rightarrow$	$2{:}\neg R$
(BR7)	$1{:}\neg R, 2{:}M$	$\rightarrow$	$2{:}\neg R$	(BR8)	$1{:}\neg R, 2{:}\neg M$	$\rightarrow$	$2{:}\neg L$
(BR9)	$2{:}L \wedge M$	$\rightarrow$	$1{:}L$	(BR10)	$2{:}L \wedge \neg M$	$\rightarrow$	$1{:}R$
(BR11)	$2{:}R \wedge M$	$\rightarrow$	$1{:}R$	(BR12)	$2{:}R \wedge \neg M$	$\rightarrow$	$1{:}L$
(BR13)	$2{:}\neg L \wedge M$	$\rightarrow$	$1{:}\neg L$	(BR14)	$2{:}\neg L \wedge \neg M$	$\rightarrow$	$1{:}\neg R$
(BR15)	$2{:}\neg R \wedge M$	$\rightarrow$	$1{:}\neg R$	(BR16)	$2{:}\neg R \wedge \neg M$	$\rightarrow$	$1{:}\neg L$

Figure 2: Bridge rules among $Mr.1$ and $Mr.2$ beliefs.

1. For any propositional letter p, $\mathsf{add}(p)$ and $\mathsf{del}(p)$ are atomic local updates.
2. If upd_1 and upd_2 are local updates, then $\mathsf{upd}_1;\mathsf{upd}_2$ is a local update.
3. Nothing else is a local update.

A local update has a *main effect*, explicitly specified in the operator, and a derived effect, not specified in the operator. The former is the set of local updates that are necessary to keep a chain consistent with respect to the set of bridge rules. Intuitively, the main effect of $\mathsf{add}(p)$ and $\mathsf{del}(p)$ is that p is set to true and false respectively; the main effect of $\mathsf{upd}_1;\mathsf{upd}_2$, is the effect of the sequential execution of upd_1 and upd_2. The following definition should clarify this intuition. In the following, l denotes a literal, i.e. a propositional letter, or the negation of a propositional letter. l^c denotes the complement of the literal l, i.e. p, if l is $\neg p$, and $\neg p$ if l is p.

Definition 6 (Main Effect). Let upd be a local update, the *main effect* of upd is defined as follows:

1. $me(\mathsf{add}(p)) = \{p\}$
2. $me(\mathsf{del}(p)) = \{\neg p\}$
3. $me(\mathsf{upd}_1;\mathsf{upd}_2) = me(\mathsf{upd}_2) \cup (me(\mathsf{upd}_1) \setminus me(\mathsf{upd}_2)^c)$, where for any set of literals S, $S^c = \{l^c | l \in S\}$.

Let's comment point 3 of the previous definition. The main effect of a sequence of two updates is the union of the effects of the two updates, minus the effect of the first update that are overwritten by the second one. For example, the effect of $me(\mathsf{del}(p);\mathsf{add}(p))$ is that p is true on the final state of the context, and from the given definition is equal to $\{p\} \cup (\{\neg p\} \setminus \{\neg p\}) = \{p\}$.

Let us now focus on the definition of the derived effects of a local update. For these effects we do not give an explicit definition, we rather give three *principles* and implicitly define the update operation as the one that satisfies the principles. The principles are:

Admissibility The result of an update must satisfy a given set of bridge rules.

Consistency The result of an update must be a consistent set of epistemic states.

Necessary changes Each derived effect should be "justifiable" to be necessary in order to maintain consistency. Here by justifiable we mean that the derived effect is a logical consequence of the main effect and the unchanged part.

The definition of the update operation is obtained by rephrasing the previous principles in a more formal way.

Definition 7 (Contextual update). Let upd be a local update on the language L_i and BR be a set of bridge rules. The *contextual update*, $i{:}\mathsf{upd}$ is a function that, for each chain $c \in C_{BR}$ returns a compatibility relation $i{:}\mathsf{upd}(c)$ containing all the chains c' such that:

1. c' satisfies the set BR of bridge rules;
2. $i{:}me(\mathsf{upd}), \Sigma(c,c') \models \Delta(c,c')$;
3. $i{:}me(\mathsf{upd}), \Sigma(c,c') \not\models i{:}\bot$.

where: $\Sigma(c,c')$, the common part of c and c', is equal to $\{i:\phi \mid c \models i:\phi \text{ and } c' \models i:\phi\}$, and $\Delta(c,c')$, the difference between c and c', is equal to $\{i:\phi \mid c \not\models i:\phi \text{ and } c' \models i:\phi\}$.

Example 4. Consider Example 3, with the initial situation formalized by chain (3). Suppose that $Mr.1$ updates his beliefs as described in the example by applying the local update $\mathsf{del}(L);\mathsf{add}(R)$. Consider the chains (5) and (6) both satisfy condition 1 and 3 of Definition 7 on the set of bridge rules BR1–16, but only (6) is acceptable as (5) does not satisfy condition 2 of Definition 7. Indeed, we have that the change of M into $\neg M$ in 2 is not justifiable in terms of logical consequence as $1{:}\neg L, 1{:}R, 2{:}\neg R, 2{:}L \not\models 2{:}\neg M$.

Notice that there are cases where there is no c' that satisfies point 1-3 of Definition 7 and therefore $i{:}\mathsf{upd}(c)$

is not defined. In this case update is not allowed since it conflicts with one of the three principles.

Example 5. Let $c = \{\{p\},\{q\}\}$ be a chain on two languages L_1 and L_2 containing the propositions $\{p\}$ and $\{q\}$ respectively. Consider the single bridge rule $1:p \rightarrow 2:q$. The set $2:\mathsf{del}(q)(c)$ is the empty set. Indeed if q is set to false in 2, in order to satisfy the rule, some change should be performed in 1. On the other hand, there is no way to infer such a derived change in 1. Notice the absence of bridge rules that go from 2 to 1.

3.2 PROPERTIES

In this section we compare the contextual update operator with two of the main references in belief update: the foundation paper [Katsuno and Mendelzon, 1991], and the review paper [Herzig, 1998]. Contextual update satisfies the four basic desiderata for belief revision described in [Herzig, 1998].

Proposition 1. *If* $i:\mathsf{upd}(c)$, *is defined, then:*

1. *Consistency:* $i:\mathsf{upd}(c) \not\models i:\bot$.
2. *Syntax Independence: If* $i : me(\mathsf{upd}_1) = i : me(\mathsf{upd}_2)$, *and* c *and* c' *are equivalent (i.e. they satisfy the same formulas), then* $i:\mathsf{upd}_1(c) = i:\mathsf{upd}_2(c')$ *are equivalent.*
3. *Success:* $i:\mathsf{upd}(c) \models i:me(\mathsf{upd})$.
4. *Minimal changes: For all* $c' \in i:\mathsf{upd}(c)$, *then for all* c'' *such that* $\Delta(c,c') \subset \Delta(c,c'')$, $c'' \notin i:\mathsf{upd}(c)$.

Belief revision is usually defined for formulas, while here we consider only literals. In order to compare our approach with the traditional AGM postulates [Alchourrón et al., 1985] of belief revision and the KM postulates [Katsuno and Mendelzon, 1991] for belief update, we need to generalize our update operators to the case of complex formulas $i:\phi$.

For an i-formula ϕ, let $\mathrm{DNF}_\phi = \{S_1,\ldots,S_k\}$ be the disjunctive normal form of ϕ, and let upd_ϕ be the set of update operations defined as follows: for any $S = \{p_1,\ldots,p_k,\neg q_1,\ldots,\neg q_k\} \in \mathrm{DNF}_\phi$, upd_ϕ contains the update operation $\mathsf{upd}_S = \mathsf{add}(p_1);\ldots;\mathsf{add}(p_n);\mathsf{del}(q_1);\ldots;\mathsf{del}(q_n)$. The generalized update operation $C \circ i:\phi$ for a compatibility relation C is defined as follows:

Definition 8. For any compatibility relation C and any i-formula ϕ, the compatibility relation C *updated with* $i:\phi$, denoted by $C \circ i:\phi$, is defined as:

$$\bigcup_{\mathsf{upd}\in\mathsf{upd}_\phi} \left(\bigcup_{c\in C} i:\mathsf{upd}(c)\right)$$

Notice, again, that $C \circ i:\phi$, is defined only when $i:\mathsf{upd}(c)$ is defined for all $c \in C$. When $C \circ i:\phi$ is defined, the update operation satisfies the seven KM postulates [Katsuno and Mendelzon, 1991]. For lack of space we omit the details of the comparison.

When the knowledge base is split into different contexts, there are other new interesting properties of the update operator concerning the relation between the different contexts. We show that our update operation has the following important property:

> **Locality of effects:** *If the knowledge base* i *is independent from the knowledge base* j, *then an update in* i *does not affect* j.

To prove this property, we need to define when a context j is *independent* from a context i. To this purpose, we introduce the notion of *bridging path*. Given a set of bridge rules BR a *bridging path* from i to j is a sequence of bridge rules $br_1,\ldots br_n$ of BR, such that the consequence of br_n is in j and the indexes of every premise of br_k $(1 \leq k \leq n)$ is equal to i or to the index of the consequence of a br_h with $h < k$. Intuitively a bridging path contains the sequence of bridge rules that one has to apply to derive, in context j, the consequences of the assumptions in the context i. We say that j is independent from i if there is no bridging path from i to j.

Proposition 2 (Locality of effects). *If* BR *does not contain a bridging path from* i *to* j, *then* $C \models i:\phi$ *if and only if* $C \circ j:\psi \models i:\phi$.

4 COMPUTING CONTEXTUAL UPDATES

In this section, we propose a sound and complete procedure, based on Natural Deduction [Prawitz, 1965], that computes the chains in the compatibility relation $i:\mathsf{upd}(c)$. This procedure builds a tree of literals in contexts, called *propagation tree*. A propagation tree for $i:\mathsf{upd}(c)$ is build by starting form the main effect of upd in i, by applying bridge rules to determine derived effects, and by making assumptions that certain literals in contexts are not changed. With no loss of generality we define the procedure w.r.t a set of bridge rules BR of the form $i_1:\phi_1,\ldots i_n:\phi_n \rightarrow i:\psi$. where ϕ_k $(1 \leq k \leq n)$ and ψ are disjunctions of literals.

Definition 9 (Propagation tree). A propagation tree is a finite tree T, such that each node of T is associated either with a set of expressions of the form $i:l$ and $i:l^h$, or with FAIL or OK. A branch t of T is *closed* if it contains a node with OK, or with FAIL,

otherwise t is *open*. T is *closed* if all its branches are closed, otherwise T is *open*. t is a *valid* branch if it is closed and contains a node labelled with OK. T is *valid* if it contains a *valid* branch.

Expressions of the form $i:l^h$ are called *hypothesis* or *assumptions*. As a point of notation, we use $i:l^{(h)}$ to denote both $i:l$ and $i:l^h$. We also need the concept of *local entailment*. A set of indexed literals S (i.e. a set of formulas of the form $i:l$) *locally entails* another set of indexed literals S', if, for each $i:l \in S'$, either $i:l^{(h)} \in S$ or $i:\bot \in S$.

Definition 10 (Update propagation tree). Let c be a chain satisfying a set BR of bridge rules, and upd a local update on L_i. The *propagation tree* for $i:\mathsf{upd}$ in c is the tree T, built starting from the root node labelled with $i:me(\mathsf{upd})$ and applying the following rules to any open branch t of T in this order:

R0 (Derive False): If $i:l^{(h)} \in t$ and $i:\neg l^{(h)} \in t$ and $i:\bot \notin t$, then extend t with $i:\bot$;

R1 (Close with fail): If $i:\bot \in t$ for all $i \in I$ and t does not contain any assumption, then extend t with FAIL;

R2 (Apply bridge rules): Let $\{br_1, \ldots br_n\} \subseteq BR$ be the set of bridge rules such that t locally entail the premises of br_k, and does not locally entails the consequence of br_k. For any $1 \leq k \leq n$ let $i_k:\phi_k$ be the consequence of br_k. Extend t with the nodes labelled with the set $\{i_1:l_1 \ldots, i_n:l_n\}$ for any choice of $l_k \in \phi_k$, with $1 \leq k \leq n$.

R3 (Discharge assumptions): Let N be a node of t containing a literal $j:l^h$ and such that all the branches containing N contain also $j:\bot$. Let $\{i_1:l_1^h, \ldots, i_n:l_n^h\}$ be the set of hypothesis occurring in these branches between N and the first occurrence of $j:\bot$ below N. For each of such nodes N, extend t with the brother nodes $N_1, \ldots, N_n$ of N, where N_k is labelled with the following set:

$$\{j:l^c, i_1:l_1^h, \ldots, i_n:l_n^h\}$$

R4 (Make minimal assumptions): Let $br \in BR$ be a bridge rule. If one of the two following conditions hold:

1. t contains at least one of the indices appearing in br and there is a minimal set of literals H, such that $c \models H$ and $t \cup H$ locally entails the premises of br; and t does not locally entail the consequence of br;
2. there is a minimal set of literals H, such that $c \models H$ and $c \cup H$ locally entails $j:\bot$, not already in t.

then, extend t with the leaf node labelled with $\{i:l^h | i:l \in H\}$.

R5 (Close branch): If no more rules can be applied extend t with FAIL if $i:\bot \in t$ for all $i \in I$ and with OK otherwise.

Definition 11 (Chains of a tree). Let T be a closed popagation tree for $i:\mathsf{upd}(c)$ and t a valid branch of T. The *chain* corresponding to t, denoted by c^t, is obtained by changing c as follows: for all $i:l \in t$ and for all $m \in c_i$, if l^c in m, then replace l^c with l in m. The set of chains of T is given by the set of chains c_t corresponding to a valid branch t of T, that satisfies the bridge rules BR.

Example 6. Consider once again Example 1: suppose to have the initial situation where both $Mr.1$ and $Mr.2$ see no balls and $Mr.2$ doesn't see $Mr.1$ in front of him. The chain that formalizes this situation is $c = \langle \square\square, \square\square\square \rangle$. Suppose now that $Mr.1$ updates his beliefs by adding a ball to the left slot of the box. The minimal effects on the beliefs of $Mr.2$ is that he believes that there is a ball in the right slot of the box. Therefore the resulting chain should be $c' = \langle \bullet\square, \square\bullet\square \rangle$. This unique solution is computed in the propagation tree shown in figure 3.

We start with the root node containing the main effect of the update $1:L$. The first applicable rule is R4 and we therefore make the assumption $2:\neg M$, which allows us to locally entail (BR2) and to obtain the node $2:R$ by rule R2. Again the first applicable rule is R4. Possible minimal assumptions are: $2:\neg R$ (which allows us to locally entails BR16 and to obtain $2:\bot$), $2:\neg L$ (which allows us to locally entails BR14), $1:\neg L$ (which allows us to locally entails BR6 and to obtain $1:\bot$) and $1:\neg R$ (which allows us to locally entails BR8). In order to simplify figure 3 we omit this last branch, which has structure similar to the one obtained starting from the assumtion $2:\neg L$. Then, we can go throught the tree to discharge the assumptions, until no more rules can be applied except R5.

The bridge rules BR1-16 of Example 1, have a very special form. Namely all the consequences of the bridge rules are literals. Furthermore in this example we don't have any local constraint, i.e. some constraint that must be satisfied inside a single context. In the next example we slightly enrich the example in order to show the behavior of the algorithm in more complex cases.

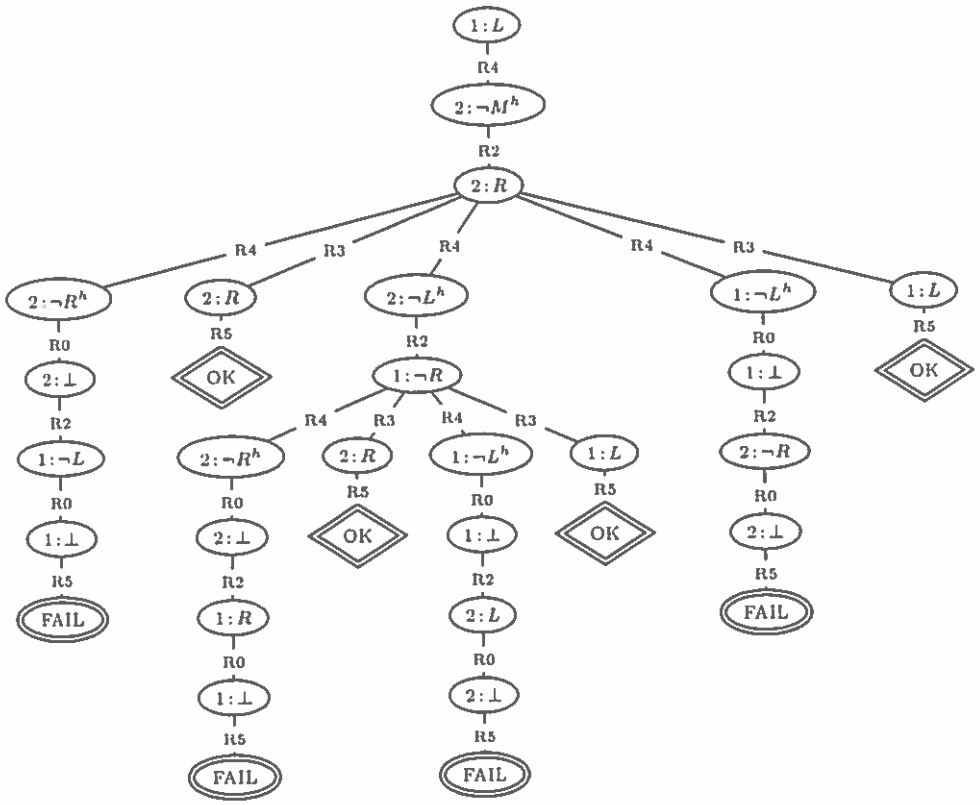


Figure 3: Propagation tree for $1{:}\mathsf{add}(L)(c)$

Example 7. Let us modify Example 1, by introducing the local constraint establishing that $Mr.1$ can see at most one ball, formalized by the bridge rule with no premises BR0, shown in the box below.

(BR0)	$\rightarrow$	$1{:}\neg L \vee \neg R$
(BR1') $1{:}L$	$\rightarrow$	$2{:}M \supset L$
(BR2') $1{:}L$	$\rightarrow$	$2{:}\neg M \supset R$
(BR3') $1{:}R$	$\rightarrow$	$2{:}M \supset R$
(BR4') $1{:}R$	$\rightarrow$	$2{:}\neg M \supset L$

We also allow $Mr.1$ to affect $Mr.2$'s belief about the man being in front of him. We therefore rephrase bridge rules BR1–4 with the bridge rules BR1'–BR4' of the box (the symbol $\supset$ is the implication). To simplify the construction of the tree we drop bridge rules BR5–8 and BR13–16. Consider the initial chain $c = \langle [\bullet\,|\,], [\bullet\,|\,|\,\mathrm{M}] \rangle$, formalizing the beliefs of $Mr.1$ and $Mr.2$ in the situation shown in the left box of Figure 1. Figure 4 shown the propagation tree computing the two chains in $2{:}\mathsf{del}(M)(c)$. (In order to simplify the tree we omit some branches with FAIL closure node.) Namely $c' = \langle [\,|\bullet], [\bullet\,|\,|\,] \rangle$ and $c'' = \langle [\,|\,], [\,|\,|\,] \rangle$. Notice finally that the propagation tree shown in Figure 4 produces also one partially inconsistent soluction, $c''' = \langle \emptyset, [\bullet\,|\bullet\,|\,] \rangle$. This solution describes the situation where Mr.1 reaches an inconsistent information state (he indeed believes that there is at most one ball, but from the coordination with 2 he get to know that there are two balls).

Theorem 1 (Soundness). *Let T be the update propagation tree for $i{:}\mathsf{upd}$ in c. If t is a valid branch of T, then $c^t \in i{:}\mathsf{upd}(c)$.*

Proof. Outline: We start by showing, by induction on the construction of T, that, for any node N of T, if H is the set of assumptions occurring in the nodes above N and $i{:}l$ is a literal of N, then

$$i{:}me(\mathsf{upd}), H \models i{:}l \tag{12}$$

Second, notice that, if t is a valid branch (and therefore closed), c^t satisfies all bridge rules. Indeed, by contradiction, suppose that c^t does not satisfy a bridge rule br, this means then c^t satisfies the premises of br and does not satisfy its consequence. This implies that t is not closed, as there is a set of assumptions H that, together with the literals in t, satisfies the premises of br, making the rules R2 and R4 applicable.

Finally notice that, from the one hand, $\Delta(c, c^t)$ is a subset of the literals of t and, from the other hand,

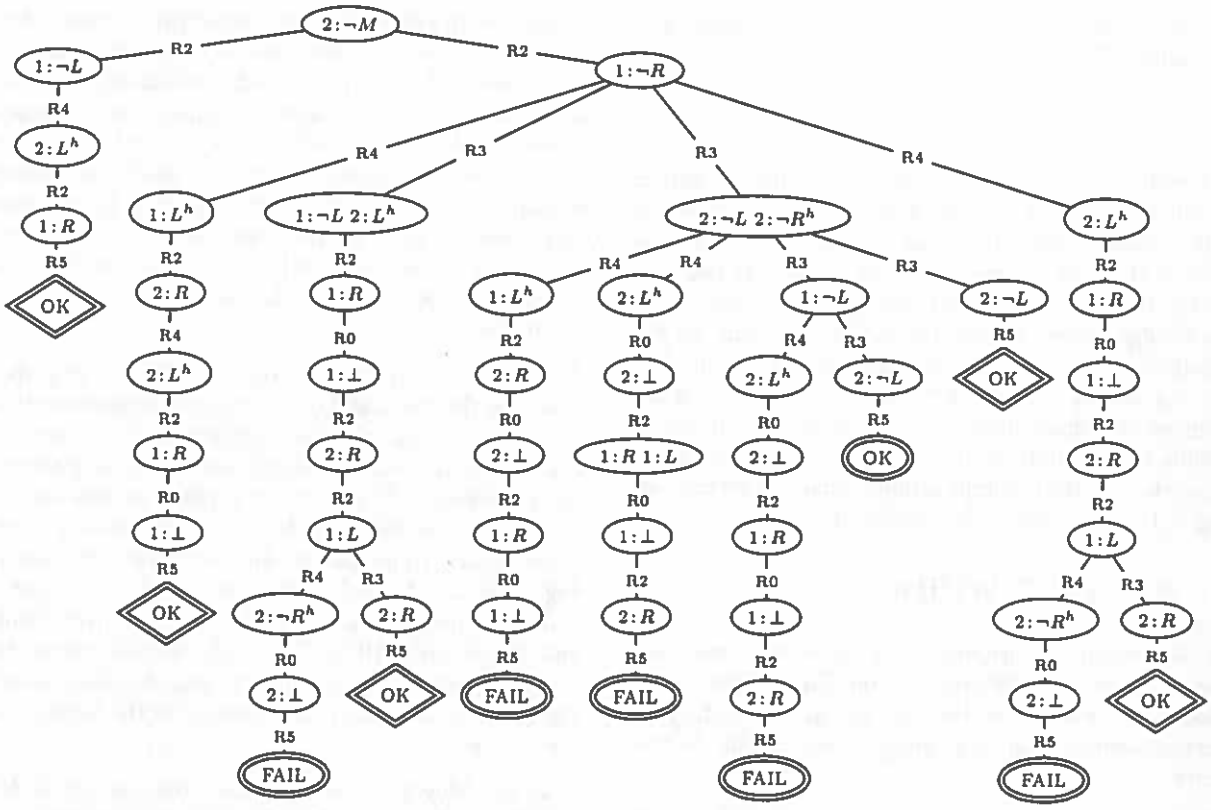


Figure 4: Propagation tree for $2{:}\mathsf{del}(M)(c)$

the set H of assumptions of t is a subset of $\Sigma(c, c^t)$. By (12) we have that:

$$i{:}me(\mathsf{upd}), \Sigma(c, c^t) \models \Delta(c, c^t) \tag{13}$$

□

Theorem 2 (Completeness). *For any $c' \in i{:}\mathsf{upd}(c)$ there is a valid branch t in the update propagation tree of $i{:}\mathsf{upd}(c)$, such that $c^t = c'$.*

Proof. Outline Let $c' \in i{:}\mathsf{upd}(c)$. We show that the propagation tree T for $i : \mathsf{upd}(c)$ contains a thread t such that $c^t = c'$.

The *discrepancy degree* of a node N of T (w.r.t. c'), denoted by $d(N)$, is equal to the number of literals occurring above or in N not satisfied by c'.

We show by induction that there is a leaf of a propagation tree with discrepancy degree equal to 0. The consequence being that the tread t from the root to this leaf, is valid and $c^t = c'$.

As a base case we have that the root has always discrepancy degree equal to 0. For the step case, suppose that N is a non leaf node at depth n with $d(N) = 0$. Let N' be a child of N with $d(N') > 0$. This implies that N' contains a literal. $j{:}l$ such that $c' \not\models j{:}l$. Consider the two cases: $j{:}l^c \in \Sigma(c, c')$ and $j{:}l^c \in \Delta(c, c')$

In the first case $j : l$ is not an assumption, and if H is the set of assumptions occurring above N then $i{:}me(\mathsf{upd}), H \models j{:}l$, and since $H \subseteq \Sigma(c, c')$, we have that $i : me(\mathsf{upd}), \Sigma(c, c') \models j : \bot$. Which is c' is inconsistent in j. This implies however that $d(N') = 0$ which contradicts the hypothesis.

In the second case $j : l^c \in \Delta(c, c')$, i.e., $j : l$ is an assumption. We distinguish the following two sub cases:

1. If $j : \bot$ appears in all the threads containing N, then by rule R3 (discharge assumptions) there is another child of N, say N'' that contains $j : l^c$, with $d(N'') = 0$.
2. If there is a thread not containing $j : \bot$, then we have the following two facts

$$i{:}me(\mathsf{upd}), \Sigma(c, c') \models j{:}l^c \tag{14}$$

$$i{:}me(\mathsf{upd}), H \not\models j{:}l^c \tag{15}$$

with $H \subset \Sigma(c, c')$. From (14) and (15) one can infer that there is a minimal set of assumptions H' in $\Sigma(c, c')$ that, if added to H can fire some new bridge rules. This implies that by rule R4,

N is extended with a node N' that contains H', with $d(N') = 0$.

□

A detailed study of the complexity of the algorithm is out of the scope of this paper, and it is part of the future work. However, we can observe that the dimension of the tree generated by the algorithm depends on the bridge rules and not from the dimension of the knowledge bases. A first (rough) upper-bound on the ramification rate of the propagation tree is equal to the maximum between m^{n-1} and 2^p (where m is the number of bridge rules, n is the maximal number of disjunctions contained in the consequence of a bridge rule, and p is the different propositional letters appearing in the premises of the bridge rules).

5 RELATED WORK

In the area of the formalisms for contextual reasoning, based either on McCarthy or on Giunchiglia's intuitions, this work is, to the best of our knowledge, the first attempt to add some form of "animation" to contexts.

With respect to the well developed area of belief revision and update, we do not introduce new concepts. Our target is complementary as we are more concerned with study of the interactions among local updates in a distributed setting. The approach proposed in this paper is analogous to the MCD (Minimal Change with Maximal Disjunctive Inclusion [Zhang and Foo, 1996]) based on PMA (Possible Models Approach [Winslett, 1988]). The main difference being that we deal with a set of interdependent knowledge bases and not with a unique knowledge base. A second difference is that the knowledge base, in this paper, is represented semantically as a set of propositional models and not as a set of formulas in certain cases this is not possible.

The approaches to the problem of revision of multi-agent beliefs (MABR) described in [Kfir-Dahav and Tennenholz, 1996], [van der Meyden, 1994], [Liu and Williams, 1999] and [Liu and Williams, 2000] are related. [Kfir-Dahav and Tennenholz, 1996] define a framework for MABR, where agents are required to be consistent on their shared beliefs. In this approach, agents' beliefs are represented by a Knowledge Base on a propositional language L. The primitive propositions of L are partitioned in two sets, *private propositions* and *shared propositions*. Agents must agree on their beliefs on shared propositions. When an agent observes a new fact ϕ, he updates his beliefs accordingly (by any revision strategy); this might involve changes both on private and shared propositions, and, therefore, updates to other agents beliefs. Such behavior is captured by our approach. Intuitively, a proposition p shared by two agents, i and j, can be represented by the bridge rules, $i{:}p \rightarrow j{:}p$ and $j{:}p \rightarrow i{:}p$. In addition, our approach allows a more fine grained coordination that enables us for instance, to represent asymmetric shared propositions with the bridge rule $i{:}p \rightarrow j{:}p$. These propositions represent the fact that changes propagate in the direction "from i to j" and not in the other direction.

The proposal of [Liu and Williams, 2000] for MABR is based on the paradigm of shared knowledge structure. In this paradigm, the knowledge base of an agent i, KB_i, is partitioned in private knowledge and accessible knowledge. The latter is further partitioned depending on the different degree of accessibility to the other agents. In particular, for each agent j the knowledge base $K_{ass}(i,j) \subseteq KB_i$ is the portion of i's accessible knowledge by agent j. Similarly to [Kfir-Dahav and Tennenholz, 1996], $K_{ass}(i,j)$ can be represented in our approach by means of bridge rules, namely the fact $p \in K_{ass}(i,j)$ corresponds to the bridge rule $i{:}p \rightarrow j{:}p$.

[van der Meyden, 1994] proposes the concept of *Mutual Belief Revision*. Mutual belief revision is the process by which a set of agents revise their beliefs and their mutual beliefs as a consequence of a common observation. At a given time, each agent i is associated with its beliefs, and the beliefs about other agents, (represented by a set of trees isomorphic to a K45 Kripke structure) and a local update function ρ_i. The local update function is supposed to be common knowledge. Roughly, the global update function is defined in terms of iterative composition of local update functions. This work has at least two points in common with our approach. First, update is defined on semantics structures, rather than on set of sentences. Second, global update is the result of the combination of local updates (performed by each agent). The main difference between our approach and the one proposed in [van der Meyden, 1994] is that we have an explicit language to express coordination between agents: bridge rules. Coordination in [van der Meyden, 1994] is instead an underlying assumption which states that local revision functions of each agent are mutually believed. This means that the updates performed by agent i on the beliefs of agents j are the same as that performed by agent j itself. As noticed in [Liu and Williams, 1999], this is not a reasonable hypothesis in an heterogeneous distributed system.

6 CONCLUSIONS

In this paper we propose a first attempt to define an update operator for contexts. The proposal relies on the formalisms of local model semantics proposed in [Ghidini and Giunchiglia, 2001]. Technically, we propose an implicit definition of update in context, and provide an algorithm that computes it. We furthermore compare our approach with the current trends in the area of Multi Agent Belief Revision. This work is motivated by providing well founded algorithms for update in distributed and heterogeneous knowledge bases, in the absence of a global representation of the knowledge. Future work will involve discovering an optimized version of the algorithm, studying its complexity, and implementing it in a distributed agent based architecture.

Acknowledgements

We would like to thank Marco Aiello, Roberta Ferrario, Fausto Giunchiglia, and Mark Carman for useful feedback and proof readings. This work is partially supported by the project Trentino Experience funded by the Provincia Autonoma di Trento. Finally, we would like to thank the anonymous referees for the helpful comments.

References

[Alchourrón et al., 1985] Alchourrón, C. E., Gardenfors, P., and Makinson, D. (1985). On the logic of theory change: Partial meet contraction and revision functions. In *Journal of Simbolic Logic*, volume 50, pages 510–530.

[Benerecetti et al., 2000] Benerecetti, M., Bouquet, P., and Ghidini, C. (2000). Contextual Reasoning Distilled. *Journal of Theoretical and Experimental Artificial Intelligence*, 12(3):279–305.

[Buvač et al., 1995] Buvač, S., Buvač, V., and Mason, I. A. (1995). Metamathematics of Contexts. *Fundamentae Informaticae*, 23(3):412–419.

[Buvač and Mason, 1993] Buvač, S. and Mason, I. A. (1993). Propositional logic of context. *Proc. of the 11th National Conference on Artificial Intelligence.*

[Ghidini and Giunchiglia, 2001] Ghidini, C. and Giunchiglia, F. (2001). Local models semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127(2):221–259.

[Giunchiglia, 1993] Giunchiglia, F. (1993). Contextual reasoning. *Epistemologia, special issue on I Linguaggi e le Macchine*, XVI:345–364. Short version in Proceedings IJCAI'93 Workshop on Using Knowledge in its Context, Chambery, France, 1993, pp. 39–49. Also IRST-Technical Report 9211-20, IRST, Trento, Italy.

[Giunchiglia and Serafini, 1994] Giunchiglia, F. and Serafini, L. (1994). Multilanguage hierarchical logics, or: how we can do without modal logics. *Artificial Intelligence*, 65(1):29–70. Also IRST-Technical Report 9110-07, IRST, Trento, Italy.

[Herzig, 1998] Herzig, A. (1998). Logics for belief base updating. In Dubois, D., Gabbay, D., Prade, H., and Smets, P., editors, *Handbook of defeasible reasoning and uncertainty management*, volume 3 - Belief Change, pages 189–231. Kluwer Academic Publishers.

[Katsuno and Mendelzon, 1991] Katsuno, H. and Mendelzon, A. (1991). On the difference between updating a knowledge base and revising it. In Allen, J. F., Fikes, R., and Sandewall, E., editors, *KR'91: Principles of Knowledge Representation and Reasoning*, pages 387–394. Morgan Kaufmann, San Mateo, California.

[Kfir-Dahav and Tennenholz, 1996] Kfir-Dahav, N. E. and Tennenholz, M. (1996). Multi-agent belief revision. In Shoham, Y., editor, *Theoretical Aspects of Rationality and Knowledge: Proceedings of the Sixth Conference (TARK 1996)*, pages 175–196. Morgan Kaufmann, San Francisco.

[Liu and Williams, 1999] Liu, W. and Williams, M. A. (1999). A framework for multi-agent belief revision, part i: The role of ontology. In *Proceedings 12th Australian Joint Conf. on Artificial Intelligence*, volume 1747 of *Lecture Notes in Computer Science*, pages 168–179, Sydney, Australia. Springer.

[Liu and Williams, 2000] Liu, W. and Williams, M. A. (2000). A framework for multi-agent belief revision, part ii: A layered model and shared knowledge structure. In *Linköping Electronic Articles in Computer and Information Science*, volume 5(21).

[Malheiro and Oliveira, 2000] Malheiro, B. and Oliveira, E. (2000). Solving conflicting beliefs with a distributed belief revision approach. In *Proceedings of the Int. Joint Conf. IBERAMIA'2000/SBIA'2000*, pages 146–155, S. Paulo, Brazil.

[McCarthy and Buvač, 1998] McCarthy, J. and Buvač, S. (1998). Formalizing Context (Expanded

Notes). In Aliseda, A., van Glabbeek, R., and Westerståhl, D., editors, *Computing Natural Language*, volume 81 of *CSLI Lecture Notes*, pages 13–50. Center for the Study of Language and Information, Stanford University.

[Prawitz, 1965] Prawitz, D. (1965). *Natural Deduction - A proof theoretical study*. Almquist and Wiksell, Stockholm.

[Serafini and Ghidini, 2000] Serafini, L. and Ghidini, C. (2000). Context Based Semantics for Information Integration. In Bonzon, P., Cavalcanti, M., and Nossum, R., editors, *Formal Aspects of Context*, volume 20 of *Applied Logic Series*. Kluwer Academic Publishers. Also IRST-Technical Report 9609-02, IRST, Trento, Italy.

[Serafini and Giunchiglia, 2000] Serafini, L. and Giunchiglia, F. (2000). ML Systems: A Proof Theory for Contexts. Technical Report 0006-01, ITC-IRST, Trento, Italy. To appear in the Journal of Logic Language and Information.

[van der Meyden, 1994] van der Meyden, R. (1994). Mutual belief revision (preliminary report). In Doyle, J., Sandewall, E., and Torasso, P., editors, *Principles of Knowledge Representation and Reasoning: Proc. Fourth International Conference (KR '94)*, pages 595–606.

[Winslett, 1988] Winslett, M. (1988). Reasoning about action using a possible models approach. In Smith, R. and Mitchell, T., editors, *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 429–450, Menlo Park, California. AAAI Press.

[Zhang and Foo, 1996] Zhang, Y. and Foo, N. Y. (1996). Updating knowledge bases with disjunctive information. In *AAAI*, pages 562–568.

Complexity Issues II

Many-sorted Preference Relations

Matteo Cristani
Department of Computer Science, University of Verona
Strada le Grazie, 15, Verona (Italy)
cristani@sci.univr.it

Abstract

The expression of qualitative preference order between objects is a rather common fact in human knowledge. Ordering by preference is a process which naturally distinguishes sorts of objects, so that we would not assert "This wonderful dessert is better than Notre Dame's Cathedral" (except in the case in which we want to depict a metaphor).

Though the above two observations are simple, no current model of preference order exists which takes in account both aspects, namely which sets up comparability of objects only when they are of the same object sort.

We provide a partial order preference relation in presence of sorts. In particular, we name $\mathcal{POIS}$-2 the *relation algebra* generated by a preference relation with two sorts, $\mathcal{POIS}$-3 the algebra of preference with finitely many sorts (more than two) and $\mathcal{POIS}$ the algebra with infinitely many sorts. We analyze the behaviour of these algebras in terms of compositions. Finally we prove that the problem of deciding the consistency of a set of binary constraints on the algebras $\mathcal{POIS}$-2 and $\mathcal{POIS}$ are polynomially solvable on deterministic machines, whilst that problem is NP-complete in $\mathcal{POIS}$-3. We provide, for $\mathcal{POIS}$ a constraint-based algorithm for deciding consistency which is $O(n^3)$. We also prove that the general path-consistency solution method can be applied in $O(n^3)$ too.

1 Introduction

In the recent literature of Artificial Intelligence a significant effort has been spent by many scholars of the area, about the problem of reasoning in presence of preference order expressed in a logical context. This is an interesting research topic for many reasons:

- The expression of preferences is a significant component of natural language. Several sentences in texts and conversations contain expressions of the form *I like this church more than the old one where we were used to go*, or of the form *I definitely prefer spaghetti with pesto with respect to hamburgers*;
- Many applications in Information Technology involve expression of preferences, like options of programs, structuring of `.ini` files in many operating systems, query optimization in Database Management Systems;
- In various cases, in particular in the applications of the Internet to the electronic commerce, the formal expressions of preference can help the user when making a purchase decision.

Though some results have been obtained by the investigations about preference order in logical context, in the current literature of Artificial Intelligence the problem of representing and reasoning about preference order in a qualitative constraint-based framework has not yet been dealt with. This lack of effort is mainly due, to our knowledge, to the absence of similar studies in related fields, in particular in the researches on Decision Support Systems, and Decision Theory, the investigations involving the representation of preference relations within Artificial Intelligence.

Why should a qualitative constraint-based framework be useful in practice? We provide in below three exam-

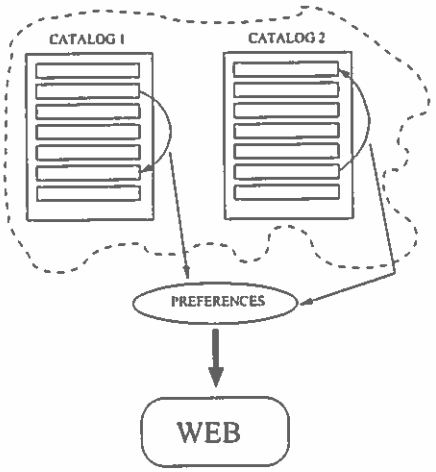


Figure 1: A schema of an electronic merchandiser based on the expression of preference in a qualitative framework.

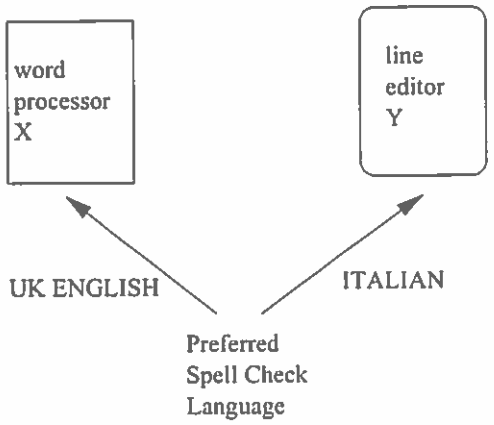


Figure 2: A schema of an system preference setup application based on the expression of preference in a qualitative framework.

ples, one in the context of natural language, one in the context of software engineering (the design of a semi-automatic tool for setting preferences in an operating system), and finally one in the context of electronic commerce (the development of a web site for helping users in making purchases).

Example 1 (*Natural Language Processing*)
A natural language interface of a complex application, for instance a data warehouse of commercial information, is used for getting and then processing user data for marketing purposes. In particular the user is asked to fill out many forms where he/she expresses preferences amongst products in various lists. The interface makes questions which guide the user through the product lists. The transition between lists is made by means of preference expressions. Clearly the fact that someone prefers the product a to the product b is not enough to establish, for instance, that another person is going to set up the same preference order. However, such a system can be able to represent "general preference order" in the sense of an "average taste", and this can be used to manage further initiative, for instance the order in which products are presented to the user of such a context (we might name such a component of the application we sketched above an *electronic merchandiser*). A schema of the above described application is presented in Figure 1. □

Example 2 (*System preferences*)
An application is used to set up the preferences in an operating system. When expressing preferences each user defines the relations he/she intends to express amongst application settings as in, for instance, "When using the Word Processor X, I want the default language of spell check to be UK English, whilst when using the line editor Y, I want it to be Italian". Preferences of each user, define, for each type of settings, a total order. In fact, we have second-choice decisions like "If there is no UK English spell check component , then use American English". The above sketched application is pictorially represented in Figure 2. □

Example 3 (*Electronic commerce*)
In an on-line auction assistance system the main help consists in using data of the preference orders established by the user for making purchases. The system integrate aesthetic and pragmatic considerations which the user can make about what he/she likes and what he/she prefers with respect to what he/she does not like of prefer less. The system express all these preferences in qualitative constraints. Making a decision, for the user, involves three steps: setting the preference orders, choosing the best available option, check the logical consistency of the decision he/she made, and if not consistent, getting back to the first step. The help provided to the user is in what the application makes by itself: the consistency checking process. □

The paper is organized as follows. Section 2 examines related work in the area; Section 3 introduces some terminology and basic definitions of relation algebra and constraint satisfaction processing; Section 4 presents the model of preference relation we intro-

duce and discusses the basic property of the obtained relation algebra, whose structural and computational behavior is deeply discussed in Section 5; finally Section 6 takes some conclusions about the nature of the research effort we have documented here.

2 Related work

In the recent past a significant attention has been posed by several scholars of the AI community to various problems of preference expression, as for Knowledge Representation or for Automated Reasoning, and in the specific case for Belief Representation and Revision and the development of Decision Support Theory. The model of preference as adopted in the current literature , is the Bayesian Decision Theory. For general information on that model refer to [Holtzman, 1989, Tversky and Kahneman, 1986, Keeney and Raiffa, 1976, Raiffa, 1968].

Within this research direction three main lacks of the theory have been dealt with:

- The theory is naturally metric, and in many cases it does make sense to express preferences, with purely qualitative information [Boutilier, 1994, Tan and Pearl, 1994];
- The computation of preference elicitation [Chajewska et al., 1998, Ha and Haddawy, 1998, Ha and Haddawy, 1997], which presuppose the notion of preference order [Karzanov and Kachiyan, 1991] is not possible in classical Bayesian Decision Theory;
- Utility independence [Koksalan and Sagala, 1995, Bacchus and Grove, 1996] and specifically the representation of multiple criteria preference reasoning [Bacchus and Grove, 1997, Basu et al., 1998] requires a slight modifications of the general assumptions of the theory. In particular we need to take in account the notion of partial order as model for preference relation, in opposition to the traditional Bayesian model, which is a total order.

We explicitly focus on the problem of qualitative representation. We are going to incorporate both preference orders and the notion of distinguishable mutually exclusive sorts, which is some sense considered for multiple criteria decisions. Our framework is thus very general, and though various efforts have been spent to address, for instance, the three above sketched problems, our problem has not yet been specifically analyzed.

The formalization of non classical preference relations, in particular of those models in which preference can be viewed as a semi-order have been deeply investigated in Economics and Operation Research recent literature. It is clear, since the early investigations like [Vincke, 1980] that classical models of preference, which presuppose that a preference relation can be modeled as partial order in which the incomparability relation represents the indifference threshold, cannot be applied to those situations (very common in practice) in which the decision-maker clearly hesitates between indifference and strict preference [Vincke, 1981, Roy and Vincke, 1984, Roy and Vincke, 1987]. Therefore various scholars have proposed approaches based on the introduction of two [Vincke, 1988] or more than two [Vincke, 2000] thresholds. Those representations have been investigated also from a computational point of view [Ngo et al., 2000].

This research direction is one of the most interesting extensions to classical partial (or total) order as model of preference defined in literature, but it still does not take in consideration the problem of comparison in preference determined by the existence of sorts. This fact has been taken in some account in those investigations which incorporate qualitative reasoning techniques in the representation of preference as a multiple criteria decision problem (for a general reference to the classical formulation of this problems see [Roubens and Vincke, 1985]. For more recent references look at [Pirlot and Vincke, 1997, Boyssou and Vincke, 1998]).

The three applications we have sketched in Section 1 are not the only one, and there exists a major research topic to which the preference order knowledge is specifically relevant: the medical diagnosis (see, for instance, [Kassier, 1994] for a medical science reference, or [Ha and Haddawy, 1998] for an AI perspective). However, for the medical applications the qualitative knowledge representation of preferences is insufficient, since abstracting away from metric details can bring to a wrong decision in many cases.

Another important field of application for preference relations, where many interesting results have been obtained recently is Database Research, and in particular query optimization [Chomicki, 2002]. In this field preference relations are used to obtain data in the best possible order and to extract query incrementally starting with the most likely object to be selected.

For the representation of the preference relations, as claimed above, we shall provide a purely qualitative framework. We shall specifically employ three *relation*

algebras which have been studied by Düntsch, amongst others, in [Düntsch, 1991]. Some of the complexity results we shall provide in this paper can also partly be derived by a rather exhaustive investigation about the complexity of relation algebras which are significant in different applications of Knowledge Representation and Reasoning, which has been carried out by Hirsch [Hirsch, 1997].

3 Terminology and definitions

The definitions relative to relation algebras are taken from Lyndon's papers [Lyndon, 1950, Lyndon, 1956] as a general reference. For technical details have also been useful [Düntsch, 1991, Hirsch, 1997, Tarski, 1941].

Definition 1 *A structure*

$$\mathcal{R} = \langle \mathcal{A}, \sqcup, \sqcap, -, \bot, \top, \circ, ^\smile, 1' \rangle,$$

where:

- $\mathcal{A}$ *is a set whose elements are called* relation symbols*;*
- $\sqcup, \sqcap, \circ$ *are binary operations on* $\mathcal{A}$*;*
- $-, ^\smile$ *are internal functions on* $\mathcal{A}$ *(unary operations);*
- $\bot, \top, 1'$ *are elements of* $\mathcal{A}$*;*

and the following axioms hold:

A.1 $\langle \mathcal{A}, \sqcup, \sqcap, \bot, \top \rangle$ *is a Boolean algebra;*

A.2 $\langle \mathcal{A}, \circ, ^\smile, 1' \rangle$ *is an involuted monoid;*

A.3 The conditions $(a \circ b) \circ c = 0$*;* $(a^\smile \circ c) \circ b = 0$ *and* $(c \circ b^\smile \circ a = 0$ *are equivalent for any* a*,* b*,* c *in* $\mathcal{A}$*.*

is called a relation algebra.

Given a relation algebra $\mathcal{R}$, if $\mathcal{A}$ is a set of relations on a nonempty set U, and the following facts hold:

- The Boolean operations are the usual set-theoretic union, intersection and complement;
- $\circ$ is relative multiplication of relations, so that the set $R \circ S$ is formed by all those pairs (x, y) such that there exists a z in U and $(x, z) \in R$, $(z, y) \in S$;
- $\bot$ is the empty set;
- $\top = U \times U$;
- $R^\smile = \{(y, x) | (x, y) \in R\}$;
- $1'$ is the identity on U $(1' = \{(x, x) | x \in U\}$;

then $\mathcal{R}$ is said to be a *set relation algebra* or a *proper relation algebra.* A proper relation algebra $\mathcal{R}$ is said to be *integral iff* the domain of each relation R of $\mathcal{R}$ coincides with the range of R and equals the universe of the relations. Formally

$$\mathtt{dom}R = \mathtt{ran}R = U$$

For a relation algebra $\mathcal{R}$ a set S such that $S \subseteq \mathcal{A}$ is said to be a *(compositional) generator set* for $\mathcal{R}$ *iff* any relation in $\mathcal{R}$ can be written by means of an expression using only relations in S and the operations of composition, intersection and converse.

When a relation algebra $\mathcal{R}$ is isomorphic to some proper (set) relation algebras we say that $\mathcal{R}$ is representable.

Consider a relation algebra, $\mathcal{R}$, whose support is $\mathcal{A}$. For any finite set of *variables* V, to be defined on a domain for one interpretation of $\mathcal{R}$, if any, a function from $V \times V$ to $\mathcal{A}$ is called a Network of constraints on $\mathcal{R}$, and indicated by

$$N = \{((x_1, y_1), r_1), ((x_2, y_2), r_2) \ldots ((x_n, y_n), r_n)\}$$

The problem of establishing whether for a given Network of constraints on a relation algebra $\mathcal{R}$, we can assign to the variables x_i and y_i, values which makes true the logical conjunction of the assertion of $r_i(x_i, y_i)$ is named *Network Satisfiability Problem (NSP)* on $\mathcal{R}$. An assignment making true all these conjuncts is called a *solution* for the NSP.

For a relation algebra three distinct formulations of the NSP can be obtained, which all start with the general definition of the framework. Given a relation algebra $\mathcal{R}$, we make the following definitions:

- **AR-NSP** (*Algebra Representability and NSP*) is the problem of establishing whether $\mathcal{R}$ is representable and for one given representation there exists a value assignment making the problem satisfiable for that representation;
- **G-NSP** (*General NSP*) is the problem of proving the existence of a representation of the algebra where the NSP has a solution, knowing a priori all the possible representations for the algebra;
- **M-NSP** (*Model NSP*) is the problem of proving the existence of a solution for the NSP, for one given representation.

The latter is henceforth called the *Constraint Satisfaction Problem* (CSP). For the sake of simplicity, when referred to a proper relation algebra $\mathcal{R}$, we intend the CSP to be referred to the domain in which the relations of $\mathcal{R}$ itself are defined. For such cases we speak of $\mathcal{R} - SAT$ referring to the generic CSP on $\mathcal{R}$ when the representation to which we refer is the identical representation. In other terms, given an proper algebra of relations $\mathcal{R}$, $\mathcal{R}$-SAT denotes the M-NSP on the identical representation.

We name *Domain of preference*, a set D, which is the union of an enumerable set of object sorts, $\{D_1, D_2, \ldots D_n \ldots\}$, each of those is fully ordered by a dense order (and therefore infinite).

4 Preference Orders and sorts

The notion of preference order is very intuitive, however, we still do not have a formal model in the current literature of Artificial Intelligence which takes in account both the notion of absolute preference and the idea of comparability as limited to objects of the same sort.

We make here some general assumptions:

- In one case, we assume the sorts to be infinitely many; this does not correspond to the reality but is very likely to be used as a simplification in any practical application; we then shortly analyze the behavior of our algebra for finitely many sorts, and distinguish both algebraically and computationally the case in which the sorts are two and the case in which they are finitely many but more than two;
- We restrict sorts to be an enumerable set of enumerable sets, for the sake of simplicity in the proofs of some basic relevant results;
- We assume the sorts to be densely and fully ordered. This is again a simplification, which does not restrict practical use. Let us recall that a direct consequence of this assumption is that a sort is an infinite set.

Definition 2 *Given an enumerable collection of disjoint sets $\mathcal{D} = \{D_1, D_2, \ldots, D_n, \ldots\}$, where any elements D_i of the collection is an infinite enumerable set on which a dense total order $\geq_{D_i}$ is defined, the relation $\trianglerighteq$ defined on $D = \bigcup_i D_i$, by*

$$x \trianglerighteq y \equiv_{def} \exists D_i \in \mathcal{D} | x, y \in D_i \wedge x \geq_{D_i} y$$

is said to be a preference relation *on D.*

The interpretation we give to $x \trianglerighteq y$ is that x is (weakly) preferable to y. The following claim holds.

Lemma 1 *A preference relation is a proper weak partial order.*

Proof Trivially $\trianglerighteq$ is a partial order, since it is reflexive, antisymmetric and transitive. To show instead that $\trianglerighteq$ is proper, note that for any pair of distinct elements D_i and D_j in $\mathcal{D}$ a pair of elements in D, (x, y), such that x is in D_i and y in D_j is neither in the order $x \trianglerighteq y$ nor in the order $y \trianglerighteq x$. ■

Based on a preference relation we can develop four relations: the equality, the strong partial order and its converse relation, the incomparability relation. These correspond to the following definitions:

$$\begin{array}{lll} \mathsf{EQ}xy & \equiv_{def} & (x \trianglerighteq y) \wedge (y \trianglerighteq x) \\ \mathsf{PF}xy & \equiv_{def} & x \trianglerighteq y \wedge \neg(y \trianglerighteq x) \\ \mathsf{PF}^{-1}xy & \equiv_{def} & y \trianglerighteq x \wedge \neg(x \trianglerighteq y) \\ \mathsf{INC}xy & \equiv_{def} & \neg(x \trianglerighteq y) \wedge \neg(y \trianglerighteq x) \end{array}$$

A second way, which will result important in the algebraic analysis of the next section is based on the two relations, which can be defined when we establish a bijective correspondence among the classes of the domain of preference[1] . In particular we can exhibit a correspondence ϑ_{ij} between D_i and D_j, which maps each element of D_i in one element of D_j. Once we have such a correspondence we can introduce the total order relation $\succ$, based on the strong relation $>_{D_i}$, derived from $\geq_{D_i}$, as follows:

$$x \succ y \equiv_{def} (x \in D_i \wedge y \in D_j \wedge \vartheta_{ij}(x) >_{D_i} y)$$

Moreover we can define the relation $\sim$, as follows

$$x \sim y \equiv_{def} \exists D_i \in \mathcal{D}[x, y \in D_i]$$

The preference relations will thus be defined as follows:

$$\begin{array}{lll} \mathsf{EQ}xy & \equiv_{def} & x = y \\ \mathsf{PF}xy & \equiv_{def} & x \succ y \wedge x \sim y \\ \mathsf{PF}^{-1}xy & \equiv_{def} & y \succ x \wedge x \sim y \\ \mathsf{INC}xy & \equiv_{def} & \neg(x \sim y) \end{array}$$

The above formalization will henceforth called the *unified representation.*

The four relations EQ, PF, PF^{-1}, INC are clearly mutually exclusive and jointly exhaustive. The composition

[1] This correspondence can be defined thanks to the enumerability of the sorts, or better, thanks to the equivalence of the transfinite cardinalities of the sorts (all the sorts being continuous would permit the same correspondence).

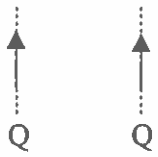


Figure 3: A schema of the $\mathcal{POIS}$-2 algebra.

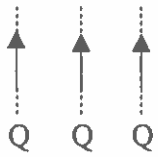


Figure 4: A schema of the $\mathcal{POIS}$ and $\mathcal{POIS}$-3 algebra.

of EQ with one of the three other relations R or of one of the relations R with EQ is trivially equal to R.

The other nine compositions of the relations depend on the number of sorts. With two sorts, the composition table will be as in Table 4. For three or more than three sorts the composition table will be as in Table 4.

A schema of the behaviour of the algebras $\mathcal{POIS}$-2, and $\mathcal{POIS}$ or $\mathcal{POIS}$-3 (which are compositionally equivalent) named by Düntsch respectively *Type 1* and *Type 2* of small integral relation algebras generated by a partial order [Düntsch, 1991], is represented in Figures 3 and 4.

We can formally prove that the composition tables mentioned above are correct.

Lemma 2 *The composition in $\mathcal{POIS}$-2 is defined by Table 1.*

Proof Adopting the unified representation we define PF as

$$\mathsf{PF}xy \equiv_{def} x \succ y \wedge x \sim y$$

and the incomparability relation as

$$\mathsf{INC}xy \equiv_{def} \neg(x \sim y).$$

Without loss of generality we can assume that each of the sorts involved is the set of rational numbers $\mathbb{Q}$, and that any object O in the domain of preference is identified with one pair $O = \langle \sigma, v \rangle$, where σ is a value between 0 and 1 and identifies the sort to which O belongs. The relation $\sim$ is the identity over $\{0, 1\}$ and the relation $\succ$ is the strict majority on rational numbers. Henceforth the proof is a straightforward application of the properties of the above defined objects. ■

Lemma 3 *The composition in $\mathcal{POIS}$-3 and $\mathcal{POIS}$ is defined by Table 2.*

Proof Analogous definition of PF and INC given for $\mathcal{POIS}$-2 are valid in $\mathcal{POIS}$-3 and $\mathcal{POIS}$. However the model we have to use is not based on the use of one pair of rational number sets, but on one triple of those numbers. We therefore define an object O in the domain of preference for $\mathcal{POIS}$-3 by means of $O = \langle \sigma, v \rangle$ where σ varies on $\{0, 1, 2\}$ and v is a rational number. For $\mathcal{POIS}$ we have $O = \langle \sigma, v \rangle$ where σ varies on $\mathbb{N}$. In both cases we assume $\sim$ to be the identity over $\{0, 1, 2\}$ and $\mathbb{N}$ respectively, and $\succ$ to be the strict majority on rational numbers. Again, the proof is a straightforward application of the properties of the above defined objects. ■

We can now claim the below theorem, where, for the sake of clarity, we denote by $A^{\cup}$ the closure by union of A, namely the set formed by the union of any subset of A.

Theorem 1 *Given a domain of preference D and the relations* EQ, PF, PF^{-1}, INC *defined on D, the structure $\langle \{\mathsf{EQ}, \mathsf{PF}, \mathsf{PF}^{-1}, \mathsf{INC}\}^{\cup}, \cap, \cup, -, \circ, ^{-1}, \mathsf{EQ} \rangle$ where $\circ$ represents the composition of relations and $^{-1}$ the relation converse, is a relation algebra.*

Proof Lemmas 2 and 3 prove that $\langle \{\mathsf{EQ}, \mathsf{PF}, \mathsf{PF}^{-1}, \mathsf{INC}\}^{\cup}, \circ, ^{-1}, \mathsf{EQ} \rangle$ is an involuted monoid. $\langle \{\mathsf{EQ}, \mathsf{PF}, \mathsf{PF}^{-1}, \mathsf{INC}\}^{\cup}, \cap, \cup, - \rangle$ is trivially a Boolean algebra, thus the claim is proved. ■

The algebras introduced in Theorem 1 are named $\mathcal{POIS}$-2 when the number of sorts in the domain of preference is two, $\mathcal{POIS}$-3 when the number of sorts is finitely many n, with $n \geq 3$ and $\mathcal{POIS}$ when the sorts are infinitely many.

We can prove the following computational results.

Lemma 4 *$\mathcal{POIS}$-2-SAT is polynomially solvable on deterministic machines.*

Proof (Sketch)
We show that any NSP defined on $\mathcal{POIS}$-2 can be

	EQ	PF	PF^{-1}	INC
EQ	EQ	PF	PF^{-1}	INC
PF	PF	PF	{EQ, PF, PF^{-1}}	INC
PF^{-1}	PF^{-1}	{EQ, PF PF^{-1}}	PF^{-1}	INC
INC	INC	INC	INC	{EQ, PF PF^{-1}}

Table 1: The composition table of the $\mathcal{POIS}$-2 algebra.

	EQ	PF	PF^{-1}	INC
EQ	EQ	PF	PF^{-1}	INC
PF	PF	PF	{EQ, PF, PF^{-1}}	INC
PF^{-1}	PF^{-1}	{EQ, PF, PF^{-1}}	PF^{-1}	INC
INC	INC	INC	INC	T

Table 2: The composition table of the $\mathcal{POIS}$-3 and of the $\mathcal{POIS}$ algebra.

solved by path-consistency[2]. In fact suppose that we have proved that a set of $\mathcal{POIS}$-2 relations among variables defined on a domain of preference $\{0,1\} \times \mathbb{Q}$ is path-consistent, namely for any variable y, the relation between x and z is tighter than the composition of the relation between x and y and the relation between y and z. If this holds we can easily show how to *separate* the problem in two parts: one is obtained by leaving the relation of identity or diversity on each edge where the $\mathcal{POIS}$-2 relation and the other one by projecting a full order relation derived by the $\mathcal{POIS}$-2 relation itself. The first of these two problems is evidently a two-colourability, whilst the second one is a Point Algebra NSP (see [van Beek, 1992] for a general reference). Both these problems can be decided in $O(n^2)$. ■

Lemma 5 *$\mathcal{POIS}$-3-SAT is NP-complete.*

Proof (Sketch)
Consider the singleton subset of $\mathcal{POIS}$-3 formed by the relation INC alone. The problem P of deciding an M-NSP on the identical representation for this subset is equivalent to the problem P' of deciding whether a given graph (obtained by unlabelling edges of the network representing P) is three-colourable. Therefore, since P' is NP-complete, then $\mathcal{POIS}$-3-SAT is NP-hard.

An obvious brute-force algorithm for deciding $\mathcal{POIS}$-3-SAT can be obtained in two steps, the first consisting in exploiting all the possible combinations of labels for the nodes amongst $\{0,1,2\}$. This algorithm has a direct nondeterministic polynomial implementation in the full exploitation of combinations, which is linear in time. ■

[2]For a general reference to path-consistency techniques see [Mackworth and Freuder, 1985]. For its use in relation algebra see [Ladkin and Maddux, 1994]

EQxy	$x \succeq y \wedge y \succeq x \wedge x \sim y$
PFxy	$x \succeq y \wedge x \neq y \wedge x \sim y$
$PF^{-1}xy$	$y \succeq x \wedge x \neq y \wedge x \sim y$
INCxy	$x \not\sim y$

Table 3: The four basic relations of $\mathcal{POIS}$ represented by the relations $\succeq$, $\neq$, $\sim$ and $\neg \sim$.

5 Algebraic Analysis

Based on the ORD-Horn theory developed by Nebel and Bürckert [Nebel and Bürckert, 1995] and on the unified representation we can develop a PREF-Horn Theory. This will give out a representation to all the 16 disjunctions of basic relations of the $\mathcal{POIS}$ algebra, by means four operators: $\succeq$, $\neq$, $\sim$ and $\not\sim$.

In particular, the basic relations of $\mathcal{POIS}$ correspond to the representations of Table 3.

We define positive the literals $\succeq$ and $\sim$, and negative the literals $\neq$ and $\not\sim$.

Table 4 represents in the first column a relation of $\mathcal{POIS}$, and in the second column the form which can be obtained by representing each basic relation by the formulae of Table 3. Table 5 represents in the first column a relation of $\mathcal{POIS}$, and in the second column the form which can be obtained by representing each basic relation by the formulae of Table 3 and then applying the distributivity of $\wedge$ with respect to $\vee$. In general this gives raise to a disjunction of many PREF-Horn clauses.

Note that the second column of Table 5 only contains PREF-Horn clauses, so that we can claim the following lemma.

Lemma 6 *The $\mathcal{POIS}$ Algebra can be fully represented in the PREF-Horn Theory.*

By Lemma 6 we derive the following theorem.

Relation	Representation as disjunction of basic relations
$\{\mathsf{EQ},\mathsf{PF}\}xy$	$(x \succeq y \wedge y \succeq x \wedge x \sim y) \vee$ $(x \succeq y \wedge x \neq y \wedge x \sim y)$
$\{\mathsf{EQ},\mathsf{PF}^{-1}\}xy$	$(x \succeq y \wedge y \succeq x \wedge x \sim y) \vee$ $(y \succeq x \wedge x \neq y \wedge x \sim y)$
$\{\mathsf{EQ},\mathsf{INC}\}xy$	$(x \succeq y \wedge y \succeq x \wedge x \sim y) \vee$ $x \not\sim y$
$\{\mathsf{EQ},\mathsf{PF},\mathsf{PF}^{-1}\}xy$	$(x \succeq y \wedge y \succeq x \wedge x \sim y) \vee$ $(x \succeq y \wedge x \neq y \wedge x \sim y) \vee$ $(y \succeq x \wedge x \neq y \wedge x \sim y)$
$\{\mathsf{EQ},\mathsf{PF},\mathsf{INC}\}xy$	$(x \succeq y \wedge y \succeq x \wedge x \sim y) \vee$ $(x \succeq y \wedge x \neq y \wedge x \sim y) \vee$ $x \not\sim y$
$\{\mathsf{EQ},\mathsf{PF}^{-1},\mathsf{INC}\}xy$	$(x \succeq y \wedge y \succeq x \wedge x \sim y) \vee$ $(y \succeq x \wedge x \neq y \wedge x \sim y) \vee$ $x \not\sim y$
$\{\mathsf{PF},\mathsf{PF}^{-1}\}xy$	$(x \succeq y \wedge x \neq y \wedge x \sim y) \vee$ $(y \succeq x \wedge x \neq y \wedge x \sim y)$
$\{\mathsf{PF},\mathsf{INC}\}xy$	$(x \succeq y \wedge x \neq y \wedge x \sim y) \vee$ $x \not\sim y$
$\{\mathsf{PF}^{-1},\mathsf{INC}\}xy$	$(x \succeq y \wedge x \neq y \wedge x \sim y) \vee$ $x \not\sim y$
$\{\mathsf{PF},\mathsf{PF}^{-1},\mathsf{INC}\}xy$	$(x \succeq y \wedge x \neq y \wedge x \sim y) \vee$ $(y \succeq x \wedge x \neq y \wedge x \sim y) \vee$ $x \not\sim y$

Table 4: The 10 properly disjunctive relations of $\mathcal{POIS}$ (excluded the basic relations and $\bot$ and $\top$) represented in form of disjunctions of basic relations.

Relation	PREF-Horn representation
$\{\mathsf{EQ},\mathsf{PF}\}xy$	$x \succeq y \wedge x \sim y$
$\{\mathsf{EQ},\mathsf{PF}^{-1}\}xy$	$y \succeq x \wedge x \sim y$
$\{\mathsf{EQ},\mathsf{INC}\}xy$	$(x \succeq y \vee x \not\sim y) \wedge$ $(y \succeq x \vee x \not\sim y)$
$\{\mathsf{EQ},\mathsf{PF},\mathsf{PF}^{-1}\}xy$	$x \sim y$
$\{\mathsf{EQ},\mathsf{PF},\mathsf{INC}\}xy$	$x \succeq y \vee x \not\sim y$
$\{\mathsf{EQ},\mathsf{PF}^{-1},\mathsf{INC}\}xy$	$y \succeq x \vee x \not\sim y$
$\{\mathsf{PF},\mathsf{PF}^{-1}\}xy$	$x \neq y \wedge x \sim y$
$\{\mathsf{PF},\mathsf{INC}\}xy$	$(x \succeq y \vee x \not\sim y) \wedge$ $(x \neq y \vee x \not\sim y)$
$\{\mathsf{PF}^{-1},\mathsf{INC}\}xy$	$(y \succeq x \vee x \not\sim y) \wedge$ $(x \neq y \vee x \not\sim y)$
$\{\mathsf{PF},\mathsf{PF}^{-1},\mathsf{INC}\}xy$	$x \neq y \vee x \not\sim y$

Table 5: The 10 properly disjunctive relations of $\mathcal{POIS}$ represented as PREF-Horn Clauses.

Theorem 2 *$\mathcal{POIS}$-SAT is polynomially solvable on deterministic machines.*

$\bot$	$\neg\mathsf{e} * \mathsf{ep} * \mathsf{ep}^{-1}$
EQ	$\mathsf{ep} * \mathsf{ep}^{-1}$
PF	$\neg\mathsf{pi} * \mathsf{ep} * \neg\mathsf{e}$
PF^{-1}	$\neg\mathsf{pi}^{-1} * \mathsf{ep}^{-1} * \neg\mathsf{e}$
INC	$\neg\mathsf{e} * \neg\mathsf{pi} * \neg\mathsf{pi}^{-1}$
$\{\mathsf{EQ},\mathsf{PF}^{-1}\}$	ep^{-1}
$\{\mathsf{EQ},\mathsf{INC}\}$	$\neg\mathsf{pi} * \neg\mathsf{pi}^{-1}$
$\{\mathsf{PF},\mathsf{INC}\}$	$\neg\mathsf{pi} * \neg\mathsf{e}$
$\{\mathsf{PF}^{-1},\mathsf{INC}\}$	$\neg\mathsf{pi}^{-1} * \neg\mathsf{e}$
$\{\mathsf{PF},\mathsf{PF}^{-1}\}$	$\neg\mathsf{e} * (\mathsf{ep} \circ \mathsf{ep}^{-1})$
$\{\mathsf{EQ},\mathsf{PF}^{-1},\mathsf{INC}\}$	$\neg\mathsf{pi}^{-1}$
$\{\mathsf{EQ},\mathsf{PF},\mathsf{PF}^{-1}\}$	$\mathsf{ep} \circ \mathsf{ep}^{-1}$
$\top$	$\neg\mathsf{e} \circ \neg\mathsf{e}$

Table 6: The compositions needed for representing $\mathcal{POIS}$, based on the generator set $\mathcal{G}_{\mathcal{POIS}}$.

Proof A set of clauses in a Horn Theory can be decided in linear time by algorithms of Positive Unit Resolution (for a reference to an optimal Positive Unit Resolution algorithm see [Dowling and Gallier, 1984]). The number of clauses needed for representing a $\mathcal{POIS}$ constraint set in PF-theory is at most cubic in the number of variables, in the hypothesis that we have a number $O(n^2)$ of constraints which require one new variable for being expressed. ■

When analyzing constraint satisfaction problems, an interesting approach is the so called constraint-based algorithmic solution. We develop now an algorithm for deciding the consistency of CSP based on $\mathcal{POIS}$ relations.

The relations in $\mathcal{POIS}$ can also be represented by means of a generator set, in particular $\{\mathsf{PF},\mathsf{PF}^{-1},\mathsf{INC}\}$, $\{\mathsf{EQ},\mathsf{PF},\mathsf{INC}\}$, $\{\mathsf{EQ},\mathsf{PF}\}$. We name this set $\mathcal{G}_{\mathcal{POIS}}$. In fact, looking at Table 6, we have the compositions needed. For the sake of simplicity we represent the relations of $\mathcal{G}_{\mathcal{POIS}}$ respectively by $\neg$e, $\neg$pi and ep. Moreover we represent the intersection of relations by $*$. Based on Table 6, we can prove the following claim

Lemma 7 *The relations* $\{\mathsf{PF},\mathsf{PF}^{-1},\mathsf{INC}\}$, $\{\mathsf{EQ},\mathsf{PF},\mathsf{INC}\}$, $\{\mathsf{EQ},\mathsf{PF}\}$ *form a generator set.*

Proof Let us name $\mathcal{G}_{\mathcal{POIS}}$ the set formed by the relations $\{\mathsf{PF},\mathsf{PF}^{-1},\mathsf{INC}\}$, $\{\mathsf{EQ},\mathsf{PF},\mathsf{INC}\}$, and $\{\mathsf{EQ},\mathsf{PF}\}$. The proof simply consists in the above table, which exhibits one possible choice of relational expressions in $\mathcal{G}_{\mathcal{POIS}}$ which define the 13 remaining relations of $\mathcal{POIS}$. ■

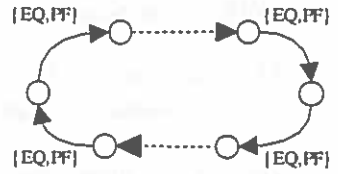


Figure 5: A $\{\mathsf{EQ}, \mathsf{PF}\}$-*cycle.*

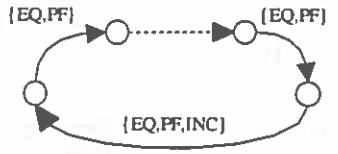


Figure 6: A *quasi* $\{\mathsf{EQ}, \mathsf{PF}\}$-*cycle.*

Note that the relation $\{\mathsf{EQ}, \mathsf{PF}\}$ is idempotent, so that in each expression of the table in the proof of Lemma 7 we can write ep^n instead of $\{\mathsf{EQ}, \mathsf{PF}\}$. When we have a pair x, y of variables on which a constraint $\{\mathsf{EQ}, \mathsf{PF}\}^n$ and a constraint $\{\mathsf{EQ}, \mathsf{PF}\}^{-1}$ are defined, we will speak of a $\{\mathsf{EQ}, \mathsf{PF}\}$-*cycle.* Conversely, when a we have two variables x and y where a constraint $\{\mathsf{EQ}, \mathsf{PF}\}$ is defined, whilst y and x are connected by $\{\mathsf{EQ}, \mathsf{PF}, \mathsf{INC}\}$ we will speak of a *quasi* $\{\mathsf{EQ}, \mathsf{PF}\}$-*cycle.* Finally, when we have two variables x and y where a constraint $\{\mathsf{EQ}, \mathsf{PF}, \mathsf{INC}\}$ and $\{\mathsf{EQ}, \mathsf{PF}^{-1}, \mathsf{INC}\}$ are defined between x and y, and a third variable z so that x and z and y and z are such that a $\{\mathsf{EQ}, \mathsf{PF}\}$ constraint is defined between them, we will speak of a $\{\mathsf{EQ}, \mathsf{PF}\}$-triangle.

In Figures 5 and 6 we respectively provide the structure of a $\{\mathsf{EQ}, \mathsf{PF}\}$-*cycle* and of a *quasi* $\{\mathsf{EQ}, \mathsf{PF}\}$-*cycle.* In Figure 7 we provide the structure of a $\{\mathsf{EQ}, \mathsf{PF}\}$-triangle.

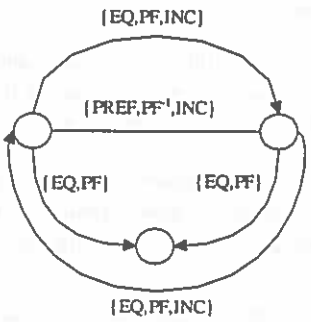


Figure 7: A $\{\mathsf{EQ}, \mathsf{PF}\}$-*triangle.*

The contradictions in $\mathcal{POIS}$ can be represented by expressions in $\mathcal{G}_{\mathcal{POIS}}$, which clearly shall contain a contradiction in $\mathcal{G}_{\mathcal{POIS}}$.

Lemma 8 *Every contradictory $\mathcal{POIS}$ constraint set represented in $\mathcal{G}_{\mathcal{POIS}}$ contains:*

- *a* $\{\mathsf{EQ}, \mathsf{PF}\}$-*cycle;*
- *a quasi* $\{\mathsf{EQ}, \mathsf{PF}\}$-*cycle;*
- *a* $\{\mathsf{EQ}, \mathsf{PF}\}$-*triangle*

and two vertices of the cycle (triangle) are connected by the relation $\{\mathsf{PF}, \mathsf{PF}^{-1}, \mathsf{INC}\}$.

Proof The contradictions in $\mathcal{POIS}$ can be computed in the below list, where the second term is the representation of the contradiction in $\mathcal{G}_{\mathcal{POIS}}$. The basic relations of $\mathcal{POIS}$ are represented as follows: $\mathsf{EQ} = \mathsf{e}$; $\mathsf{PF} = \mathsf{p}$; $\mathsf{PF}^{-1} = \mathsf{pi}$ and $\mathsf{INC} = \mathsf{n}$.

1) $\bot = \mathsf{ep} * \mathsf{ep}^{-1} * \neg\mathsf{e}$
2) $\mathsf{e} \cap \mathsf{p} = \mathsf{ep} * \mathsf{ep}^{-1} * \neg\mathsf{pi} * \neg\mathsf{e}$
3) $\mathsf{e} \cap \mathsf{pi} = \mathsf{ep} * \mathsf{ep}^{-1} * \neg\mathsf{pi}^{-1} * \neg\mathsf{e}$
4) $\mathsf{e} \cap \mathsf{n} = \mathsf{ep} * \mathsf{ep}^{-1} * \neg\mathsf{pi} * \neg\mathsf{pi}^{-1} * \neg\mathsf{e}$
5) $\mathsf{e} \cap \mathsf{ppi} = \mathsf{ep} * \mathsf{ep}^{-1} * (\mathsf{ep} \circ \mathsf{ep}^{-1}) * \neg\mathsf{e}$
6) $\mathsf{e} \cap \mathsf{pn} = \mathsf{ep} * \mathsf{ep}^{-1} * \neg\mathsf{pi} * \neg\mathsf{e}$
7) $\mathsf{e} \cap \mathsf{pin} = \mathsf{ep} * \mathsf{ep}^{-1} * \neg\mathsf{pi}^{-1} * \neg\mathsf{e}$
8) $\mathsf{e} \cap \neg\mathsf{e} = \mathsf{ep} * \mathsf{ep}^{-1} * \neg\mathsf{e}$
9) $\mathsf{p} \cap \mathsf{pi} = \mathsf{ep} * \mathsf{ep}^{-1} * \neg\mathsf{pi} * \neg\mathsf{pi}^{-1} * \neg\mathsf{e}$
10) $\mathsf{p} \cap \mathsf{n} = \mathsf{ep} * \neg\mathsf{pi} * \neg\mathsf{pi}^{-1} * \neg\mathsf{e}$
11) $\mathsf{p} \cap \mathsf{pin} = \mathsf{ep} * \neg\mathsf{pi} * \neg\mathsf{pi}^{-1} * \neg\mathsf{e}$
12) $\mathsf{pi} \cap \mathsf{n} = \mathsf{ep}^{-1} * \neg\mathsf{pi}^{-1} * \neg\mathsf{pi} * \neg\mathsf{e}$
13) $\mathsf{ep} \cap \mathsf{pn}^{-1} = \mathsf{ep} * \neg\mathsf{pi}^{-1} * \neg\mathsf{e}$
14) $\mathsf{ep}^{-1} \cap \mathsf{pn} = \mathsf{ep}^{-1} * \neg\mathsf{pi} * \neg\mathsf{e}$
15) $\mathsf{en} \cap \mathsf{ppi} = (\mathsf{ep} \circ \mathsf{ep}^{-1}) * \neg\mathsf{pi} * \neg\mathsf{pi}^{-1} * \neg\mathsf{e}$
16) $\mathsf{ppi} \cap \mathsf{n} = (\mathsf{ep} \circ \mathsf{ep}^{-1}) * \neg\mathsf{pi} * \neg\mathsf{pi}^{-1} * \neg\mathsf{e}$

It is evident that:

- The contradictions #1 and #8 are identical. These are named contradiction structure *a*;
- The contradictions #2, #3, #6 and #7 are identical. These are named contradiction structure *b*;
- The contradictions #4 and #9 are identical. These are named contradiction structure *c*;
- The contradiction #5 is named contradiction structure *d*;
- The contradictions #10, #11 and #12 are identical. These are named contradiction structure *e*;
- The contradictions #13 and #14 are identical. These are named contradiction structure *f*;

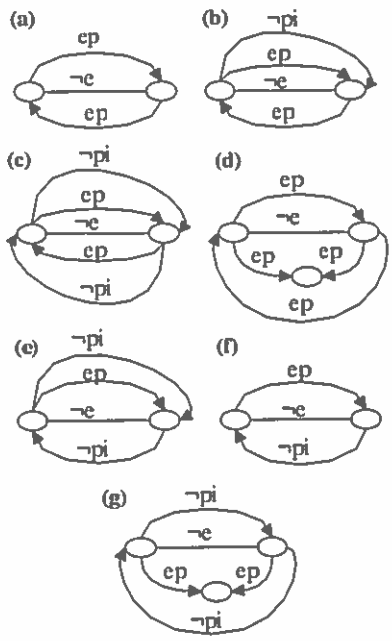


Figure 8: The seven ways of expressing contradictions in $\mathcal{POIS}$.

- The contradictions #15 and #16 are identical. These are named contradiction structure g;

The seven obtained structures of contradictions in $\mathcal{POIS}$ are shown in Figure 8.

From the above mentioned figure we derive:

- The contradictions structures a and d contain a $\{\mathsf{EQ}, \mathsf{PF}\}$-cycle with two vertices of the cycle connected by the relation $\{\mathsf{PF}, \mathsf{PF}^{-1}, \mathsf{INC}\}$;
- The contradictions structures e and f contain a quasi $\{\mathsf{EQ}, \mathsf{PF}\}$-cycle with two vertices of the cycle connected by the relation $\{\mathsf{PF}, \mathsf{PF}^{-1}, \mathsf{INC}\}$;
- The contradictions structures b and c contain both a $\{\mathsf{EQ}, \mathsf{PF}\}$-cycle and a quasi $\{\mathsf{EQ}, \mathsf{PF}\}$-cycle with two vertices of the cycle connected by the relation $\{\mathsf{PF}, \mathsf{PF}^{-1}, \mathsf{INC}\}$.
- The contradiction structure g contains a $\{\mathsf{EQ}, \mathsf{PF}\}$-triangle with two vertices of the cycle connected by the relation $\{\mathsf{PF}, \mathsf{PF}^{-1}, \mathsf{INC}\}$.

This proves the claim. ■

Based on Lemma 8 we can build Algorithm $\mathcal{POIS}$-consistency, which is described in Figure 5.

ALGORITHM $\mathcal{POIS}$-consistency

Input: A $\mathcal{POIS}$ Constraint Set X
Output: *Yes* if X is consistent, *No* otherwise

Step 1: **For** any $\{\mathsf{EQ}, \mathsf{PF}\}$ or quasi $\{\mathsf{EQ}, \mathsf{PF}\}$-cycle c
Step 2: **If** c contains $\neg$e **then Return** *No*;
Step 3: **Else If** X contains more cycles **goto** 1;
Step 4: **For** any triple $t = \{a, b, c\}$ of vertices in X
Step 5: **If** t is a $\{\mathsf{EQ}, \mathsf{PF}\}$-triangle **and** t contains $\neg$e **then Return** *No*;
Step 6: **Return** *Yes*

Figure 9: Algorithm $\mathcal{POIS}$-consistency, which decides satisfiability of a finite set of $\mathcal{POIS}$ constraints.

Cycle detection in graphs is $O(n^2)$, and detection of $\{\mathsf{EQ}, \mathsf{PF}\}$-triangles can be obviously performed in $O(n^3)$, thus we can claim the following theorem whose proof is a direct consequence of the above results on cycle and triangle detections and of Lemma 8.

Theorem 3 *Algorithm $\mathcal{POIS}$-consistency correctly decides the consistency of a $\mathcal{POIS}$ constraint set in $O(n^3)$.*

6 Conclusions and further work

We defined a new framework for reasoning about qualitative preference relations in presence of an enumerable set of distinguishable and incomparable sorts of objects.

There are several ways in which this research can be taken further. First of all we shall expand the algebras to include:

- Partially ordered sorts, in particular those models which are based on more than one threshold like {P, Q, I} - preference orders, or better {P, Q, I, J} - preference order;
- Representing inclusion of extensions for the classes, so that instead of a set of disjoint classes we have a taxonomic organization of the classes;
- Uncertainty in the representation of preference relations, by means of the introduction of one form of fuzziness in the model of preference.

We then shall integrate our model with the Qualitative Decision Theory [Tan and Pearl, 1994]and the "ceteris paribus" model of Doyle et al. [Doyle et al., 1991, Doyle and Wellman, 1994]. We will investigate the

application of our model to Electronic Commerce in the context of a large project regarding web languages which is in definition phase in our department.

Another important aspect which is going to be studied is the applicability of those models of preference studied in Database Research which are not order relations [Chomicki, 2002]. This case of study makes a big difference in what can and what cannot be said by means of a relation algebra.

Acknowledgments

I would like to thank Ivo Düntsch who kindly discussed with me some topics and read an earlier version of the paper, and Roger Maddux who pointed me references in the current literature of relation algebra. I certainly did less than I could with these helps. Many thanks also to the reviewers of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR 2002) for their useful comments. I gratefully acknowledge, in particular, reference to the economic literature about preference by one of the reviewers. Last but not least I would like to thank Jan Chomicki for pointing me some interesting aspects of the use of preference relations in Database Research.

References

[Bacchus and Grove, 1996] Bacchus, F. and Grove, A. (1996). Utility independence in a qualitative decision theory. In *Proceedings of the Fifth International Conference on Knowledge Representation and Reasoning (KR '96)*. Morgan Kaufmann.

[Bacchus and Grove, 1997] Bacchus, F. and Grove, A. (1997). Independence and qualitative decision theory. In *Working Notes of the Stanford Spring Symposium on Qualitative Decision Theory*, Stanford, CA.

[Basu et al., 1998] Basu, C., Hirsh, H., and Cohen, W. (1998). Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 714–720, Madison, WI.

[Boutilier, 1994] Boutilier, C. (1994). Toward a logic for qualitative decision theory. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR94)*, San Mateo, CA. Morgan Kaufmann.

[Boyssou and Vincke, 1998] Boyssou, D. and Vincke, P. (1998). Topics on preference modelling. *Annals of Operation Research*, 80. Preface to the Special Issue on Preference Modelling.

[Chajewska et al., 1998] Chajewska, U., Getoor, L., Norman, J., and Shahar, Y. (1998). Utility elicitation as a classification problem. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*.

[Chomicki, 2002] Chomicki, J. (2002). Querying with intrinsic preferences. In *Proceedings of the VIII. Conference on Extending Database Technology (EDBT 2002)*.

[Dowling and Gallier, 1984] Dowling, W. and Gallier, J. (1984). Linear time algorithms for testing the satifiability of propositional horn formula. *Journal of Logic Programming*, 3:267–284.

[Doyle et al., 1991] Doyle, J., Shoham, Y., and Wellman, M. (1991). A logic of relative desires (preliminary report). In *Proceedings of the 6th International Symposium on Methodologies for Intelligent Systems*, pages 16–31.

[Doyle and Wellman, 1994] Doyle, J. and Wellman, M. (1994). Representing preferences as ceteris paribus comparatives. In Hanks, S., Russel, S., and Wellman, M., editors, *Working Notes of the AAAI Spring Symposium on Decision-Theoretic Planning*, Stanford, CA.

[Düntsch, 1991] Düntsch, I. (1991). Small integral relation algebras generated by a partial order. *Periodica Mathematica Hingarica*, 23:129–138.

[Ha and Haddawy, 1997] Ha, V. and Haddawy, P. (1997). Problem-focused incremental elicitation of multi-attribute utility models. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 215–222.

[Ha and Haddawy, 1998] Ha, V. and Haddawy, P. (1998). Towards case-based preference elicitation: Similarity measures on preference structures. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 193–201.

[Hirsch, 1997] Hirsch, R. (1997). Expressive power and complexity in algebraic logic. *Journal of Logic and Computation*, 7(3):309–351.

[Holtzman, 1989] Holtzman, S. (1989). *Intelligent Decision Systems*. Addison-Wesley, Reading,Mass.

[Karzanov and Kachiyan, 1991] Karzanov, A. and Kachiyan, L. (1991). On the conductance of order markov chains. *Order*, 8(1):7–15.

[Kassier, 1994] Kassier, J. P. (1994). Incorporaing patients' preferences into medical decisions. *The New England Journal of Medicine*, 26(330):1895–1896.

[Keeney and Raiffa, 1976] Keeney, R. and Raiffa, H. (1976). *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Wiley and Sons, New York.

[Koksalan and Sagala, 1995] Koksalan, M. and Sagala, P. (1995). Interactive approaches for discrete alternative multiple criteria decision making with monotone utility functions. *Management Science*, 7(41):1158–1171.

[Ladkin and Maddux, 1994] Ladkin, P. and Maddux, R. (1994). On Binary Constraint Problems. *Journal of the ACM*, 41(3):435–469.

[Lyndon, 1950] Lyndon, R. (1950). The representation of relational algebras. *Annals of Mathematics*, 51(3):707–729.

[Lyndon, 1956] Lyndon, R. (1956). The representation of relational algebras II. *Annals of Mathematics*, 63(2):294–307.

[Mackworth and Freuder, 1985] Mackworth, A. and Freuder, C. (1985). The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25(1):65–74.

[Nebel and Bürckert, 1995] Nebel, B. and Bürckert, H. J. (1995). Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *Journal of the ACM*, 42(1):43–66.

[Ngo et al., 2000] Ngo, A., Tsoukias, A., and Vincke, P. (2000). A polynomial time algorithm to detect pqi interval order. *International Transactions on Operation Research*, 7:609–623.

[Pirlot and Vincke, 1997] Pirlot, M. and Vincke, P. (1997). *Semiorders*. Kluwer. 187 pages.

[Raiffa, 1968] Raiffa, H. (1968). *Decision Analysis*. Addison-Wesley, Reading,MA.

[Roubens and Vincke, 1985] Roubens, M. and Vincke, P. (1985). *Preference Modelling*. Number 250 in Lecture Notes in Economics and Mathematical Systems. Springer Verlag. 94 pages.

[Roy and Vincke, 1984] Roy, B. and Vincke, P. (1984). Relational systems of preference with one or several pseudo-criteria: new concepts and new results. *Management Science*, 30(11).

[Roy and Vincke, 1987] Roy, B. and Vincke, P. (1987). Pseudo-orders: definition, properties and numerical representation. *Mathematical Social Science*, 14(2).

[Tan and Pearl, 1994] Tan, S. and Pearl, J. (1994). Qualitative decision theory. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 928–932.

[Tarski, 1941] Tarski, A. (1941). The calculus of relations. *The Journal of Symbolic Logic*, 6(3):73–89.

[Tversky and Kahneman, 1986] Tversky, A. and Kahneman, D. (1986). Rational choice and the framing of decisions. *Journal of Business*, (59):S251–S278.

[van Beek, 1992] van Beek, P. (1992). Reasoning about qualitative temporal information. *Artificial Intelligence*, 58(1):297–321.

[Vincke, 1980] Vincke, P. (1980). Linear utility functions in semiordered mixture spaces. *Econometrica*, 48(3).

[Vincke, 1981] Vincke, P. (1981). Preference modelling: a survey and an experiment. *Operational Research*.

[Vincke, 1988] Vincke, P. (1988). P, Q, I - preference structures. In *Nonconventional preference relations in decision making*. Springer Verlag.

[Vincke, 2000] Vincke, P. (2000). {P, Q, I, J} - preference structures. In *Preference and Decisions under incomplete knowledge*. Springer Verlag.

From preference representation to combinatorial vote

Jérôme Lang
IRIT
Université Paul Sabatier, Toulouse (France)
lang@irit.fr

Abstract

We introduce the notion of combinatorial vote, where a group of agents (or voters) is supposed to express preferences and come to a common decision concerning a set of non-independent variables to assign. We study two key issues pertaining to combinatorial vote, namely preference representation and the automated choice of an optimal decision. For each of these issues, we briefly review the state of the art, we try to define the main problems to be solved and identify their computational complexity.

1 Introduction

For a few years, AI researchers have been concerned with interaction, cooperation or negociation within agent societies. For these problems, it often occurs that the set of all feasible states has a very large size, due to its combinatorial nature. For this reason, research has been done so as to develop representation languages aiming at enabling a succint representation of the description of the problem, without having to enumerate a prohibitive number of states. Recently, languages based on propositional logic have been proposed for some multi-agent problems, namely, combinatorial auctions [8] and automated negociation [39].

In this paper we focus on *combinatorial vote*. Combinatorial vote is located within the larger class of *group decision making* problems. Each one of a set of agents (called *voters*) initially expresses her preferences on a set of alternatives (called *candidates*), these preferences are then aggregated so as to identify (or *elect*) an acceptable common alternative in an automated way (without negociation). Formulated as such, this can be identified as a *vote problem*. Vote problems have been investigated by resarchers in social choice theory (see [29] for an overview) who have studied extensively all properties of various families of voting rules, up to an important detail: candidates are supposed to be listed explicitly (typically, they are individual or lists of individuals, as in political elections), which assumes that they should not be too numerous.

In this paper, we focus on the case where the set of candidates has a *combinatorial structure*, i.e., is a Cartesian product of finite value domains for each one of a set of variables: this problem will be referred to as *combinatorial vote*. In this case, the space of possible alternatives has a size being exponential in the number of variables and it is therefore not reasonable asking the voters to rank or evaluate on a utility scale all alternatives.

Consider for example that the voters have to agree on a common menu to be composed if a first course dish, a main course dish, a dessert and a wine, with a choice of 6 items for each. This makes 6^4 candidates. This would not be a problem if the four items to choose were independent from the other ones: in this case, this vote problem over a set of 6^4 candidates would come down to four independent problems over sets of 6 candidates each, and any standard voting rule could be applied without difficulty. Things become more complicated if voters express dependencies between items, such as "I would like to have risotto ai funghi as first course, except if the main course is a vegetable curry, in which case I would prefer smoked salmon as first course", "I prefer white wine if one of the courses is fish and none is meat, red wine if one of the courses is meat and none is fish, and in the remaining cases I would like equally red or white wine", etc.

Since the preference structure of each voter cannot be expressed explicitly by listing all candidates, what is needed is a compact *preference representation language*. Such preference representation languages have been developed within the KR community; they are often build up on propositional logic, but not always (see for instance utility networks [1] [23] or valued con-

straint satisfaction [35]); they enable a much more concise representation of the preference structure, while preserving a good readability (and hence a proximity with the way agents express their preferences in natural language).

Therefore, the first parameter to be fixed, for a combinatorial vote problem, is the language for representing the preferences of the voters. In Section 3, we recall different logical representation languages proposed so far, we discuss their relevance to combinatorial vote, and discuss their computational complexity.

In Section 4, we investigate the problem of electing a candidate; by "candidate"; we mean here an assignment of a value to each of the variables involved in the problem. We study the computational complexity of the different problems obtained from the choice of a given representation language and a given voting rule. In Section 5 we evoke other important issues in combinatorial vote and we mention some of its potential applications.

2 Preliminaries

In this paper, $\mathcal{L}$ is a propositional language built upon a finite set of propositional variables VAR, the usual connectives, and the symbols $\top$ (tautology) and $\bot$ (contradiction). A *literal* is a propositional variable or its negation. $W = 2^{VAR}$ is the set of all interpretations for VAR. Elements of W are denoted $\vec{x}$, $\vec{y}$ etc. If $\varphi \in \mathcal{L}$ then $Mod(\varphi)$ denotes the set of the models of φ and $Var(\varphi)$ is the set of propositional variables mentioned in formula φ.

The complexity results we give in this paper refer to some complexity classes which we now briefly recall (see [32] for more details). Given a problem A, we denote by $\overline{\text{A}}$ its complement. We assume the reader familiar with the classes P, NP and coNP and we now introduce the following classes.

- NP(3) is the class of all languages L such that $L = L_1 \cap (L_2 \cup L_3)$, where L_1 and L_2 are in NP and L_3 in coNP. The canonical NP(3)-complete problem is SAT-SAT-UNSAT: $\langle \varphi_1, \varphi_2, \varphi_3 \rangle$ is a positive instance of SAT-SAT-UNSAT if and only if φ_1 is consistent and (φ_2 is inconsistent or φ_3 is consistent). Theses classes are members of the *Boolean hierarchy* [10].
- $\Delta_2^p = \mathsf{P}^{\mathsf{NP}}$ is the class of all languages that can be recognized in polynomial time by a Turing machine equipped with an NP oracle, where an NP oracle solves whatever instance of a problem NP in unit time. $\Theta_2^p = \Delta_2^p[\mathcal{O}(log\ n)])$ is the class of all languages that can be recognized in polynomial time by a Turing machine using a number of NP oracles bounded by a logarithmic function of the size of the input data.
- $\Sigma_2^p = \mathsf{NP}^{\mathsf{NP}}$ 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.
- PSPACE is the class of all languages recognizable by a (deterministic or not) Turing machine working with polynomial space.

Note that the following inclusions hold:
$\mathsf{NP}, \mathsf{coNP} \subseteq \mathsf{NP(3)} \subseteq \Theta_2^p \subseteq \Delta_2^p \subseteq \Sigma_2^p \subseteq \mathsf{PSPACE}$

3 Logical representation of preference

In this Section we are concerned with the preferences of a *single voter* over a finite set of candidates $\mathcal{X}$.

3.1 Preference structures

What is a preference structure? In other terms, what is the mathematical model underlying the preferences that an agent has concerning a set of alternatives? There is not a unique answer to this important question that has been discussed for long by researchers in decision theory. One may, roughly speaking, distinguish two main families (not disjoint, however) of preference structures:

- a *cardinal preference structure* consists of an evaluation function (generally called *utility*) $u : \mathcal{X} \to Val$, where Val is a set of numerical valuations (typically, $\mathbb{N}$, $\mathbb{R}$, $[0,1]$, $\mathbb{R}^+$ etc.).
- a *relational preference structure* consists of a binary[1] relation R on $\mathcal{X}$. R is generally supposed to be a weak order (i.e., a reflexive and transitive relation); in this case the preference structure R is *ordinal*[2], and we then note $\geq$ instead of R. $\geq$ is not necessarily complete (some candidates may not be comparable by a given agent). We note

[1] More generally, this relation may be fuzzy so as to enable representing intensities of preference. We omit this eventuality in this paper.

[2] See [13] for an extensive discussion on the limits of ordinality in decision making under uncertainty, multicriteria decision making and social choice.

$\vec{x} > \vec{y}$ for $\vec{x} \geq \vec{y}$ and not $(\vec{y} \geq \vec{x})$ (strict preference of $\vec{y}$ over $\vec{x}$), and $\vec{x} \sim \vec{y}$ for $\vec{x} \geq \vec{y}$ and $\vec{y} \geq \vec{x}$ (indifference).

Note that any cardinal preference induces an ordinal preference, namely from a utility function u we may define the complete weak order $\geq_u$ defined by $\vec{x} \geq_u \vec{y}$ iff $u(\vec{x}) \geq u(\vec{y})$.

The *explicit representation* of a preference structure consists of the data of all candidates with their utilities (for cardinal preferences) or the whole relation[3] R (for ordinal preferences). These representations have a spatial complexity in $\mathcal{O}(|\mathcal{X}|)$ and $\mathcal{O}(|\mathcal{X}|^2)$, respectively, i.e., in $\mathcal{O}(2^{|VAR|})$ and $\mathcal{O}(2^{|VAR|^2})$.

3.2 Principles of logical representation of preferences

We now assume that $\mathcal{X}$ has a combinatorial nature, namely, $\mathcal{X}$ is a set of possible assignments of each of a certain number of variables to a value of its domain: $\mathcal{X} = D_1 \times ... \times D_n$, where D_i is the set of possible values for variable v_i; the size of $\mathcal{X}$ is exponentially larger than the size of the input. Because explicit representations are unreasonable for such problems. the AI community has developed several preference representation languages that escape this combinatorial blow up. Such languages are said to be *factorized*, or *succinct*, because they enables a much more concise representation of preference structures than explicit representations. In the rest of the paper, for simplicity reasons we make the following two important hypotheses:

1. the representation languages considered are *propositional*, i.e., each v_i is a binary variable: $D_1 = ... = D_n = \{0, 1\}$. Propositional languages are not only compact, but also particularly *expressive*, thanks to the expressive power of logic, and therefore close to intuition (ideally, a preference representation language should be easily obtained from its specification in natural language by the agent); moreover, propositional logic benefits from many well-worked algorithms.

2. the set of possible decisions (candidates) $\mathcal{X}$ is identical to the set of physically realizable worlds (by the agents), $Mod(K)$, where K is a propositional formula expressing a set of integrity constraints[4]. This strong assumption assumes that there is *no uncertainty* on the initial state of the world nor on the effects of actions (and this is known by all agents). It is important to keep in mind the strong distinction between K, which represents *knowledge*, and the formulas that represent *goals* (confusing both notions leads to "wishful thinking" [37]).

Now we briefly survey different logical preference representation languages. For each of these languages we discuss its computational complexity. Since the problem at hand is not to *reason about knowledge* but to *decide from preferences*, the important problems are not quite the same as for logical languages for knowledge representation. In particular, the role of inference is smaller, whereas the problems that are particularly important are:

Given a logical specification GB (goal base) of the preferences of an agent ...

- *... and two candidates $\vec{x}$ and $\vec{y}$, determine if $\vec{x} \geq_{GB} \vec{y}$* : COMPARISON;
- *... and a candidate $\vec{x}$, determine if $\vec{x}$ is optimal for $\geq_{GB}$*: NON-DOMINANCE;
- *... and a property represented by a formula ψ, determine if there exists an optimal candidate satisfying ψ* : CAND-OPT-SAT.

3.3 A synthetic survey and some complexity results

An intuitive way for an agent to express his preferences consists in enumerating a set of *goals* (or *desires*), each of which is represented by a propositional formula, possibly with extra data such as weights, priorities, contexts or distances. From now on, G_i and C_i denote propositional formulas and α_i is a positive number. GB is a "goal base" and u_{GB} (resp. $\geq_{GB}$) denotes the utility function (resp. the preference relation) induced by GB.

Some of the complexity results we give are not quite new: indeed, although the complexity of preference representation languages has not been studied in a specific way, several of the problems below are very close to similar to problems of *knowledge* representation, and although the relevant problems are not quite the same whether it is a matter of knowledge or preference, some complexity results established in the former context can induce easily similar results for the latter one.

For the results that can be easily derived from other results, we will give the appropriate reference(s); in

[3] or possibly a subset of R from which R is drawn by transitive closure – this is a detail since this does not enable escaping the combinatorial blow up.

[4] This hypothesis that all models of K are deterministically reachable from the initial situation comes down to say that there exists an individual action $a_{\vec{x}}$ for each $\vec{x} \in Mod(K)$, whose unique possible consequence is $\vec{x}$.

the absence of such references, they are new.

3.3.1 Brute propositional representation

The simplest, or prototypical, logical representation of preference, that we call "brute" and denote by R_{brute}, consists simply in giving a single goal (i.e. a propositional formula) G (or equivalently, a finite set of propositional formulas, interpreted conjunctively). The utility function u_G generated by G is extremely basic: for each possible world $\vec{x} \in Mod(K)$, $u_G(\vec{x}) = 1$ if $\vec{x} \models G$, $u_G(\vec{x}) = 0$ if $\vec{x} \models \neg G$. This representation, very rough since it does not enable more than a distinction between goal states and non-goal states ("binary" utility), is of little interest in practice but we will often refer to it because it provides lower bounds for complexity results.

Proposition 1 *For $R = R_{brute}$:*

1. COMPARISON *is in* P ;
2. NON-DOMINANCE *is* coNP*-complete ;*
3. CAND-OPT-SAT *is* NP(3)*-complete.*

Sketch of proof : (1) is obvious. The membership part of (2) comes from the fact that candidate $\vec{x} \in Mod(K)$ is optimal for $\geq_G$ iff $\vec{x} \models G$ or $G \wedge K$ is unsatisfiable. Lastly, there exists an optimal candidate satisfying ψ iff ψ is consistent and ($K \wedge G$ is inconsistent or $K \wedge G \wedge \psi$ is consistent), which gives the membership part of (3). Hardness comes from the following reduction from SAT-SAT-UNSAT: $\langle \varphi_1, \varphi_2, \varphi_3 \rangle$ is a positive instance of SAT-SAT-UNSAT iff φ_1 is consistent and (φ_2 is inconsistent or φ_3 is consistent). Assuming w.l.o.g. that φ_1, φ_2 and φ_3 do not share any variables, and p being a new variable, we let $K = \top$, $G = (\varphi_2 \wedge \neg p) \vee (\varphi_2 \wedge \varphi_3)$, $\psi = \varphi_1 \wedge p$.

3.3.2 Weighted goals

Let us start by defining the representation R_{card}, whose input is a multiset of goals $GB = \{G_1, ..., G_n\}$, and $u_{GB}(\vec{x}) = |\{i | \vec{x} \models G_i\}|$. This refinement of R_{brute} considers all-or-nothing but independent goals, and enables compensations. This representation can be easily generalized to R_{wg} by weighting goals with numerical valuations, to be then aggregated (see for instance [21]) :

$GB = \{\langle \alpha_1, G_1 \rangle, ..., \langle \alpha_n, G_n \rangle\}$ and

$$u_{GB}(\vec{x}) = F_1(F_2(\{\alpha_i | \vec{x} \models G_i\}), F_3(\{\alpha_j | \vec{x} \models \neg G_j\}))$$

i.e. $u_{GB}(\vec{x})$ is a function of the weights of goals satisfied by $\vec{x}$ and of the weights of the goals violated by $\vec{x}$

When utility can be considered as a relative notion rather than an absolute one (which means that only differences of utilities between candidates are relevant), it is often assumed that only the violated goals count (see for instance [21, 38]), which leads to $u_{GB}(\vec{x}) = F(\{\alpha_i | \vec{x} \models \neg G_i\}))$; F has of course to satisfy a number of desirable properties (see [21])[5]. Usual choices for F are, for instance, sum (weights are then usually called *penalties*) or maximum (which leads to possibilistic logic). The complexity of decision problems for weighted logics can easily be derived from the complexity of distance-based belief merging ([28],[19]). Assuming that the aggregation functions can be computed in polynomial time, we get the following results as byproducts from known results, especially the complexity of distance-based belief merging ([28], and especially [19]): COMPARISON is polynomial; NON-DOMINANCE is coNP-complete; CAND-OPT-SAT is in Δ_2^p, and in Θ_2^p when the set $\{u_{GB}(\vec{x}) \mid \vec{x} \in Mod(K)\}$ can be computed in polynomial time (and therefore has a polynomial size); CAND-OPT-SAT is already Θ_2^p-complete for R_{card}.

3.3.3 Prioritized goals

Let us now define $R_{\subseteq}$: given $GB = \{G_1, ..., G_n\}$, let $sat_{GB}(\vec{x}) = \{i | \vec{x} \models G_i\}$ and $nonsat_{GB}(\vec{x}) = \{1, ..., n\} \setminus sat_{GB}(\vec{x})$. The preference relation induced by GB is: $\vec{x} \geq_{GB} \vec{y}$ if and only if $sat_{GB}(\vec{x}) \supseteq sat_{GB}(\vec{y})$

This partial preorder on $\mathcal{X}$ is nothing but the Pareto ordering: indeed, a candidate $\vec{x}$ is optimal for $R_{\subseteq}$ iff there is no candidate $\vec{y} \in Mod(K)$ satisfying all goals that $\vec{x}$ satisfies and at least another one.

This representation can be generalized, considering $GB = \langle \{G_1, ..., G_n\}, > \rangle$ where $>$ is a complete weak order (priority relation) on $\{1, ..., n\}$. The most common choice (see *e.g* [18] [3]), called the *discrimin* ordering when $>$ is ranked (i.e., the complement of $>$ is complete), is the following:
$R_{discrimin}$:

$$\vec{x} \geq_{GB,>}^{discrimin} \vec{y} \text{ if and only if}$$
$$\forall i \in nonsat_{GB}(\vec{x}) \setminus nonsat_{GB}(\vec{y})$$
$$\exists j \in nonsat_{GB}(\vec{y}) \setminus nonsat_{GB}(\vec{x}) \text{ such that } j \geq i.$$

It is not hard to show that, as soon as $\vec{x} \geq_{GB,>} \vec{y}$ can be decided in polynomial time from $>$ – which is obviously the case for $R_{\subseteq}$ and $R_{discrimin}$, then COMPARISON is polynomial and NON-DOMINANCE is in coNPand CAND-OPT-SAT is in Σ_2^p. Moreover, NON-DOMINANCE

[5] However, this assumption is sometimes not made; in this case, positive preference items (goals) have to be formally distinguished from negative ones (constraints); see [4].

is coNP-complete for $R_{\subseteq}$. Lastly, from results in [30], [15] it can be easily derived that CAND-OPT-SAT is already Σ_2^p-complete for $R_{\subseteq}$.

3.3.4 Conditional logics

Each goal G_i is now attached to a *context* $C_i : GB = \{C_1 : G_1, ..., C_n : G_n\}$, and $C : G$ is interpreted as $C \Rightarrow G$ in the simplest conditional logics: $\geq$ being a complete weak order on candidates, we say that $C : G$ is satisfied by $\geq$ if and only if $Max(Mod(C), \geq) \subseteq Mod(G)$; this can be interpreted as *"ideally G if C"* [5]. This constraint does not fully determine the preference relation induced from $\mathcal{D}$. Several possibilities exist:

(1) R_{CL}: $\vec{x} \geq_{GB} \vec{y}$ iff for any $\geq$ satisfying GB we have $\vec{x} \geq \vec{y}$. $\geq_{GB}$ is a partial preoder which is much too weak (i.e., does not enable enough comparisons) to be a good candidate for preference representation[6], as it can be seen on the following example:
Let us consider the propositional language generated by two propositional variables a and b and let $GB = \{\top : a\}$. *Then, for any* $\vec{x}, \vec{y} \in \{(a,b), (a,\neg b), (\neg a, b), (\neg a, \neg b)\}$, $\vec{x} \geq_{GB} \vec{y}$ *holds if and only if* $\vec{x} = \vec{y}$ *(and therefore* $\vec{x} >_{GB} \vec{y}$ *never holds).*

(2) R_{CLZ}: $\geq^Z_{GB}$ is the preference relation, among those satisfying GB, maximizing preference world by world ([5], page 79), which is equivalent to the Z-completion of GB [33]. The obtained relation $\geq^Z_{GB}$ is much more discriminant (hence much better) than $\geq_{GB}$. Details are omitted. Complexity results for R_{CLZ} can be derived as byproducts from complexity results for conditional knowledge bases ([16], Section 3) : COMPARISON, NON-DOMINANCE and CAND-OPT-SAT are in Δ_2^p.

One drawback of R_{CLZ} is that, as for $\geq_{GB}$), a so-called "drowning effect" occurs (some goals are ignored while they should not); this can be remedied for instance by adding extra constraints expressing that violating a conditional desire induces an explicit utility loss [24]. This principle is further generalized by introducing numerical strengths and polarities in [38] [26]. We did not investigate complexity issues for these approaches.

3.3.5 Ceteris paribus preferences

In this second interpretation of conditional desires, $C : G > G'$ is interpreted by : *all irrelevant things being equal, I prefer G to G', or in more formal terms, "all irrelevant things being equal, I prefer a* $C \wedge G \wedge \neg G'$*-world to a* $C \wedge \neg G \wedge G'$*-world.* The definitions proposed in various places [14, 36, 6] differ somehow. We take as a basis the definition of [14], slightly generalized so as to take account for integrity constraints (K).

R_{CP} : the strict order $>_{GB}$ is defined as the transitive closure of the following set of strict inequalities:

$$\left\{ \vec{x} > \vec{y} \;\middle|\; \begin{array}{l} \text{there exists } C : G > G' \text{ in } GB \text{ such that} \\ 1.\ \vec{x} \models K \wedge C \wedge G \wedge \neg G', \\ 2.\ \vec{y} \models K \wedge C \wedge \neg G \wedge G' \\ 3.\ \vec{x} \text{ and } \vec{y} \text{ coincide on all variables} \\ \quad \text{not appearing in } G, G' \end{array} \right\}$$

Note that the the definition of the variables included in the "ceteris paribus" scope of $C : G > G'$ could be refined further by considering, instead of the variables not appearing G, G', the variables on which G and G' do not semantically depend ([36, 25]). This is not be considered further in this paper.

For the complexity results we consider not only the general case but also the following two restrictions:

- *GL*: each goal G_i is a conjunction of literals and $G'_i = \neg G_i$ (which corresponds to the definition in [6], restricted to propositional domains);
- *CIG*: the contexts C_i are independent from the goals, *i.e.*, for any goal base $GB = \{C_1 : G_1 > G'_1, \ldots, C_n : G_n > G'_n\}$ we have $Var(C_i) \cap Var(G_j) = Var(C_i) \cap Var(G'_j) = \emptyset$ for any $i, j = 1 \ldots n$. This condition is realized in particular when the contexts C are empty ($C = \top$).

Proposition 2 *The complexity of* COMPARISON, NON-DOMINANCE *and* CAND-OPT-SAT *for* ceteris paribus *preferences is reported on the following table (where −c means "-complete").*

	general case	*GL*	*CIG*	*GL+ CIG*
COMPAR.	PSPACE-*c*	PSPACE-*c*	NP-*c*	P
NON-DOM.	coNP-*c*	P	coNP-*c*	P
C.-O.-SAT	Σ_2^p-*c*	NP-*c*	Σ_2^p-*c*	NP-*c*

Sketch of proof:
- the somewhat surprising *PSPACE*-completeness of COMPARISON comes from reductions to and from plan existence with deterministic actions represented in STRIPS (STRIPS-PLAN-EXISTENCE); the latter problem has been shown PSPACE-complete in [9] (PSPACE-hardness is due to the fact that plans can be exponentially long). We give a reduction from STRIPS-PLAN-EXISTENCE to the *GL* restriction of COMPARISON (the reduction in the converse direction is in [6]): from any action α with the consistent set of literals *pre* as precondition and the consistent set of literals *post* as postcondition, we draw the *ceteris paribus* preference item

[6] Just to mention it, note that complexity for R_{CL} can be derived from complexity results for conditional logics [17] and is surprisingly low : CAND-OPT-SAT is NP-complete and NON-DOMINANCE coNP-complete.

$pre \setminus NEG(post) : post > (pre \cap NEG(post))$
where $pre \setminus NEG(post)$ the conjunction of all literals that are in *pre* and whose negation is not in *post*, and $pre \cap NEG(post)$ the conjunction of all literals that are in *pre* and whose negation is in *post*. The existence of a plan leading from an initial state s to a final state s' is equivalent to a strict preference of s' to s with respect to the set of c.p. preference items corresponding to the actions. Now, under the *CIG* assumption, preference items need to be "fired" only once, therefore, if $x > y$ then there is a "path" of polynomial length between x and y, hence the membership to NP.
- for NON-DOMINANCE, membership to coNPcomes from the following property: $\vec{x}$ is non-dominated if and only if for each $C_i : G_i > G'_i$ in GB such that $\vec{x} \models K \wedge C_i \wedge \neg G_i \wedge G'_i$, there is no $\vec{y}$ such that $\vec{y} \models K \wedge C_i \wedge G_i \wedge \neg G'_i$ and such that $\vec{x}$ and $\vec{y}$ coincide on all variables that do not appear in G_i and G'_i. coNP-hardness for the *CIG* restriction comes from the following reduction from UNSAT: let $K = \top$ and $GB = \{\top : \varphi \wedge new > \neg(\varphi \wedge new)\}$, where $new \notin Var(\varphi)$, and let $\vec{x}$ such that $\vec{x} \models \neg new$. Then $\vec{x}$ is undominated if and only if φ is unsatisfiable. Under *GL*, NON-DOMINANCE is equivalent to: for all $C_i : G_i > \neg G_i$ of GB we have $\vec{x} \models C_i \Rightarrow G_i$, hence the membership to P.
- for CAND-OPT-SAT, membership to, respectively, Σ_2^p and NPcomes easily from the membership of NON-DOMINANCE to, respectively, coNPand p. Σ_2^p-hardness under the *CIG* restriction comes from a reduction from base revision (or inference from maximal consistent subsets), using the fact that $\vec{x}$ is undominated for $K = \top$ and $GB = \{\top : \phi_1 > \neg\phi_1, \ldots, \top : \phi_n > \neg\phi_n\}$ if and only if $\vec{x}$ satisfies a maximal consistent subset of $\{\phi_1, \ldots, \phi_n\}$. Under *GL*+*CIG*, NP-hardness comes from the fact that when $K = \top$ and $GB = \emptyset$, there is an undominated candidate satisfying $'\varphi$ if and only if φ is satisfiable.

The surprising point is that, for the general case, the comparison problem is much more difficult than the other ones (in contrast with other representation languages). This is due to possible exponential "preference paths". Note that the connection between the comparison problem in ceteris paribus nets and STRIPS planning was first remarked in [6].

3.3.6 Distances

Let d be a (pseudo-)distance on $\mathcal{X}$, *i.e.*, a function from $Mod(K) \times Mod(K)$ to $I\!N$ such that (i) $d(\vec{x}, \vec{y}) = 0$ if and only if $\vec{x} = \vec{y}$ and (ii) $d(\vec{x}, \vec{y}) = d(\vec{y}, \vec{x})$. Distance-based logical representations of preference [21, 22, 20] are based on the intuitive idea that, when expressing a goal G, then ideally, $\vec{x}$ must satisfy G, and when it is no longer the case, then, the "further" $\vec{x}$ is from G, the less preferred $\vec{x}$.

Formally, a pair $\langle\{G\}, d\rangle$, where G is a propositional formula and d a pseudo-distance, induces the utility function

$$u_{GB}(\vec{x}) = -d(\vec{x}, G) = -\min_{\vec{y} \models G} d(\vec{x}, \vec{y})$$

When d is computable in polynomial time, one easily derives from the literature on the complexity of belief revision [15] [31] and of distance-based belief merging [19] that COMPARISON, NON-DOMINANCE and CAND-OPT-SAT are in Δ_2^p (and, in particular, Θ_2^p-complete when d is the Hamming distance).

This principle can be generalized by considering a set of goals, each of which is associated to a pseudo-distance: $GB = \{\langle G_1, d_1\rangle, \ldots, \langle G_n, d_n\rangle\}$, and $u_{GB}(\vec{x}) = F(d_1(\vec{x}, G_1), \ldots, d_n(\vec{x}, G_n))$, where F is an aggregation function. This has no strong impact on complexity.

3.3.7 Discussion

These results have a value of their own, since they enable a first comparison of propositional preference representation from a computational point of view[7]. Now, the complexity of the key problems COMPARISON and NON-DOMINANCE also have a strong impact on the rest of the paper, i.e., to the question whether these languages are suitable for combinatorial vote. Interesting languages are then (in principle) those for which either COMPARISON or NON-DOMINANCE is polynomial, namely, in addition to the brute representation: weighted or prioritized goals, and specific restrictions of ceteris paribus desires.

The problem with weighted goals is the well-known difficulty of eliciting numerical preferences from agents; as to prioritized goals, their lack of expressivity (no compensation allowed between goals) somewhat limits their range of use. "Ceteris paribus" preferences are interesting from a cognitive point of view, being rather close to human intuition and rather easy to elicit [6] [12]. However, they have a high computational complexity in the general case, and furthermore, from the point of view of expressivity, they are not very discriminant[8].

[7] Note however that what is lacking here is a study of their *representational complexity* [11] which would assess precisely their concision power (this is an interesting topic for further research).

[8] However, see [7] for an introduction of numerical utilities in ceteris paribus networks that remedies this problem, but on the other hand reintroduces the cognitive problem inherent to numerical utilities ...

4 Combinatorial vote

We now assume that a group of p agents (or voters) have to come up with a common assignment of values to dependent variables. A combinatorial vote problem consists of two steps: first, the agents express their preferences within a fixed (and common) representation language R, and second, one or several optimal candidate(s) is (are) determined automatically, using a fixed *voting rule*.

A *preference profile* consists of a preference structure (cardinal or ordinal) for each of the voters. Social choice theory (see [29] for a good introduction) defines a voting rule V as a function mapping every preference profile P to an elected candidate or a subset of $\mathcal{X}$, called the set of *elected candidates*[9]. The set of elected candidates, given a preference profile P and a voting rule V, will be denoted by $Select_V(P)$.

Research in social choice theory focuses on the properties of voting rules (and establishes for instance impossibility results) but does not care about the representation language and computational complexity, since the preferences are supposed to be expressed explicitly. However, in combinatorial vote, the representation language is an important parameter, and the computational properties of a voting rule will both depend of the rule V itself and of the representation language R chosen.

For any representation language R, one defines a *p-voters R-profile* as a collection $\mathcal{B} = \langle GB_1, ..., GB_p \rangle$ of goal bases (one for each voter), expressed in the language R, generating a profile $P = Induce_R(\mathcal{B})$, *i.e.*, depending whether preferences are cardinal or ordinal: $Induce_R(\mathcal{B}) = \{u_{GB_1}, ..., u_{GB_p}\}$ or $Induce_R(\mathcal{B}) = \{\geq_{GB_1}, ..., \geq_{GB_p}\}$.

We will now look at several voting rules that are well-known in the social choice community, and discuss these with respect to two criteria:

1. *relevance* for combinatorial vote (i.e., does the voting rule still mean something when the set of candidates has a combinatorial structure?);
2. *computational complexity.* Let V be a voting rule and R a representation language; we consider the following decision problems:
   - AMONG-WINNERS$_{V,R}$: given $\mathcal{B} = \langle GB_1, ..., GB_p \rangle$ and a candidate $\vec{x}$, determine whether $\vec{x} \in Select_V(Induce_R(\mathcal{B}))$;
   - ELECT-SAT$_{V,R}$: given $\mathcal{B} = \langle GB_1, ..., GB_p \rangle$ and a formula ψ, determine whether there is a candidate in $Select_V(Induce_R(\mathcal{B}))$ satisfying ψ.

The complexity of these problems depends a lot on the complexity of COMPARISON and NON-DOMINANCE for R. It has been seen that for some of these languages, comparison and/or non-dominance is polynomial. For the rest of the paper, we focus on these.

4.1 Aggregative rules for cardinal preferences

When preference is cardinal, i.e., $P = \langle u_1, ..., u_n \rangle$, the simplest way consists in "aggregating before comparing" (see [13]), namely: for each $\vec{x}$, the scores $u_i(\vec{x})$ are synthesized by an aggregation function $* : \mathbb{R}^p \rightarrow \mathbb{R}$ into a so-called social welfare function $u_{\mathcal{B}}(\vec{x}) = *\{u_i(\vec{x}) | i = 1 \ldots p\}$ reflecting the satisfaction of the group (see [29]). $Select_{agreg(*)}(P)$ is then the set of candidates maximizing $u_{\mathcal{B}}(\vec{x})$. Classically, the function $*$ is commutative (for guaranteeing the anonymousness property) and non-decreasing.

In Section 3 it has been seen that the languages that lead to cardinal preference structures are weighted goals and distance-equipped goals (note that for the latter the polynomiality of comparison does not hold except if some restrictions are made).

As a consequence of Section 3 and [19] it can be shown that, provided that $u_i(\vec{x})$ is polynomially computable from $\mathcal{B}$, and $*$ is polynomially computable, AMONG-WINNERS$_{agreg(*),R}$ is in coNP and ELECT-SAT$_{agreg(*),R}$ is in Δ_2^{P}.

The problem with these methods are, again, the difficulty to elicit a numerical utility function. From now on we focus on rules whose input is an ordinal structure (which may possibly have been induced from a cardinal structure).

4.2 Scoring rules

4.2.1 General principle

Scoring rules consist in translating the preference relation $\geq_i$ of voters into score functions $s_i(\vec{x})$, such that the score $s_i(\vec{x})$ of a candidate $\vec{x}$ with respect to voter i is a function of its position in the relation $\geq_i$, *only.*

[9] For the sake of simplicity we will not distinguish between what is usually called a voting rule, which selects a unique candidate, and a voting correspondance, which selects a subset of candidates; we will use the terminology "voting rule" in all cases. The rules that we will consider in the rest of the paper are strictly speaking correspondances, from which a standard rule can be defined by tie-breaking ex-aequos either by giving up neutrality or by randomly choosing one of the elected candidates.

The global score $s(\vec{x})$ of $\vec{x}$ is then computed by summing up all scores $s_i(\vec{x})$ for $i = 1$ to n. This is particularly clear in the case where $\geq$ is a *complete order* (i.e., no ex-aequos, no incomparabilities), $r_i(\vec{x})$ is the rank of $\vec{x}$ in $\leq_i$ (1 if it is the favorite candidate for i, 2 if it is the second favorite etc.), then $s_i(\vec{x})$ is a non-increasing function of $r_i(\vec{x})$.

The most common choice is the *Borda rule*. In the case of a complete order, the Borda score is defined by $s_i(\vec{x}) = |\mathcal{X}| - r_i(\vec{x})$, which can be generalized by: let $s_i(\vec{x})$ be defined from $>_i$ as the number of candidates that do not dominate $\vec{x}$, i.e.,

$$s_i(\vec{x}) = |Mod(K)| - |\{\vec{x}' \in Mod(K) | \vec{x}' >_i \vec{x}\}|$$

One may think that scoring rules translate purely ordinal preference into a numerical utility function in a rather arbitrary way, which is true; but on the other hand, these rules satisfy several desirable properties such as monotony and some forms of strategyproofnesses that are not satisfied by other rules (see [29]).

However, in the context of combinatorial vote, scoring rules present a major drawback: being based on a counting of candidates, they are extremely syntax-sensitive, *i.e.*, to the choice of propositional symbols used for representing the problem in hand, and moreover it really makes little sense to use a rule such as Borda on a set of candidates of cardinality 2^n as soon as n does not have a reasonable size. Therefore, most scoring rules are much more arbitrary in the context of combinatorial vote than in classical vote contexts. As if it were not enough, their computational complexity, in most cases, is really high, since candidate counting is required. One may thus advance the following general principle that *a voting rule that requires a counting of candidates is not suitable to combinatorial vote*. This rules out almost all scoring rules; however, a few of them (very specific) avoid this counting, such as the plurality and the veto rules, that we study now.

4.2.2 Plurality and veto

The *plurality* rule is the scoring rule obtained by taking $s_i(\vec{x}) = 1$ if and only if $\vec{x}$ is non-dominated for $\geq_i$, *i.e.*, iff there is no $\vec{y} \models K$ such that $\vec{y} >_i \vec{x}$. $Select_{\text{plurality}}(P)$ is therefore the set of candidates maximizing the number of voters for whom $\vec{x}$ is non-dominated (or, in the simple case of a complete order, the number agents that rank $\vec{x}$ in first position). It is known to be less satisfactory than many other scoring rules, but on the other hand it is syntax-insensitive and does not require any counting of candidates , therefore it is relevant for combinatorial vote. We now look at its complexity, when the preference representation language R varies.

Proposition 3 *Let R be a language for which* NON-DOMINANCE *is in* NP *or in* coNP.

1. AMONG-WINNERS$_{plurality,R}$ *is in* Θ_2^p;
2. ELECT-SAT$_{plurality,R}$ *is in* Σ_2^p;
3. *for $R = R_{brute}$,* AMONG-WINNERS$_{plurality,R_{brute}}$ *is* coNP*-complete and* ELECT-SAT$_{plurality,R_{brute}}$ *is* Θ_2^p*-complete.*

Proof sketch: (1) is easy, once noticed that $s(\vec{x})$ is bounded by n. (2) comes easily from (1). As to (3), let $\mathcal{B} = \{GB_1, ..., GB_n\}$ with $GB_i = \{G_i\}$. $s(\vec{x})$ is maximal if and only if the number of i such that $\vec{x} \models G_i$ is maximal, hence the membership of AMONG-WINNERS$_{\text{plurality},R_{brute}}$ to coNP. coNP-hardness is a consequence of coNP-hardness of NON-DOMINANCE. As to ELECT-SAT$_{\text{plurality},R_{brute}}$, both membership and hardness come straightforwardly from cardinality-maximizing base revision [31].

(3) gives a lower complexity bound since all representation languages considered in this paper generalize R_{brute}. The upper bound for ELECT-SAT is reached for instance for $R = R_{\subseteq}$: ELECT-SAT$_{\text{plurality},R_{\subseteq}}$ is Σ_2^p-complete. For R_{wg}, ELECT-SAT is in Δ_2^p.

The *veto* rule is obtained by letting $s_i(\vec{x}) = 1$ if and only if there is at least a candidate $\vec{y}$ such that $\vec{x} >_{G_i} \vec{y}$. When $>_{G_i}$ is a complete order, this comes down to count the number of voters who do not for $\geq_i$, *i.e.*, iff there is no $\vec{y} \models K$ such that $\vec{y} >_i \vec{x}$. $Select_{\text{plurality}}(P)$ is therefore the set of candidates maximizing the number of voters for whom $\vec{x}$ is non-dominated (or, in the simple case of a complete order, the number of voters who rank $\vec{x}$ in first position). Complexity results for the veto rule are similar to those for the plurality rule.

Example 1 *Let us fix $R = R_{card}$, $\mathcal{L}$ the language generated by the propositional variables $\{a, b, c\}$ and let the number of voters be 3. Let $\mathcal{B}$ be the following 4-uple of goal bases:*

$$\mathcal{B} = \langle GB_1, GB_2, GB_3, GB_4 \rangle$$

where

- $GB_1 = \{a, a \vee b\}$;
- $GB_2 = \{\neg a, b \wedge c, b \vee c\}$;
- $GB_3 = \{\neg b, a \Leftrightarrow \neg c\}$;

The preference orderings $\geq_{G_1}$, $\geq_{G_2}$ and $\geq_{G_3}$ are:

$(a,b,c), (a,b,\neg c), (a,\neg b,c), (a,\neg b,\neg c)$
$>_{G_1}$ $(\neg a,b,c), (\neg a,b,\neg c)$
$>_{G_1}$ $(\neg a,\neg b,c), (\neg a,\neg b,\neg c)$

$(\neg a,b,c)$
$>_{G_2}$ $(a,b,c), (\neg a,b,\neg c), (\neg a,\neg b,c), (\neg a,\neg b,\neg c)$
$>_{G_2}$ $(a,b,\neg c), (a,\neg b,c)$
$>_{G_2}$ $(a,\neg b,\neg c)$

$(a,\neg b,\neg c), (\neg a,\neg b,c)$
$>_{G_3}$ $(a,b,\neg c), (\neg a,b,c), (a,\neg b,c), (\neg a,\neg b,\neg c)$
$>_{G_3}$ $(a,b,c), (\neg a,b,\neg c)$

Candidates written on a same line are indifferent: for instance, for G_1, we have $(a,b,c) \sim_{G_1} (a,b,\neg c) \sim_{G_1} (a,\neg b,c) \sim_{G_1} (a,\neg b,\neg c)$, any of these four candidates being strictly preferred to both $(\neg a,b,c)$ and $(\neg a,b,\neg c)$ and so on.

The above table shows the Borda, plurality and veto scores of all candidates.

	s_1^B	s_2^B	s_3^B	s^B	s^p	s^v
(a,b,c)	8	7	2	17	1	2
$(a,b,\neg c)$	8	3	6	17	1	**3**
$(a,\neg b,c)$	8	3	6	17	1	**3**
$(a,\neg b,\neg c)$	8	1	8	17	**2**	2
$(\neg a,b,c)$	4	8	6	**18**	1	**3**
$(\neg a,b,\neg c)$	4	7	2	13	0	2
$(\neg a,\neg b,c)$	2	7	8	17	1	2
$(\neg a,\neg b,\neg c)$	2	7	6	15	0	2

Framed scores correspond to elected candidates.

4.3 Condorcet-consistent rules

4.3.1 Condorcet winner

A *Condorcet winner* (CW) for a profile P is a candidate $\vec{x}$ such that for any candidate $\vec{y} \neq \vec{x}$, there are strictly more agents preferring $\vec{x}$ to $\vec{y}$ than agents preferring $\vec{y}$ to $\vec{x}$, *i.e.*, $\forall \vec{y} \in Mod(K), |\{i | \vec{x} >_i \vec{y}\}| > |\{j | \vec{y} >_i \vec{x}\}|$.[10] This notion comes back to 1785 and it is known since then that there are profiles for which there is no CW. Importantly, when there exists a CW, it is unique.

A first problem is that the usual definition of a CW is not well-suited to combinatorial vote. This can be seen easily in the case where $R = R_{brute}$ (and is easily be generalized to more sophisticated representation languages). We easily get that $\vec{x}$ is the Condorcet winner for $\mathcal{B} = \langle G_1, ..., G_n \rangle$ if and only if the number of i such that $\vec{x} \models G_i$ is maximal *and there is no other interpretation* than $\vec{x}$ maximizing this number of voters whose goals is satisfied. This has the consequence that there almost *never* exists a Condorcet winner in practical cases, because the number of candidates being much larger than the number of voters, there are generally pairs of worlds upon which all voters are indifferent. For instance, let $\mathcal{L}$ be the propositional language generated by the variables $\{a,b,c\}$; let the numbers of voters be 4, $R = R_{brute}$, and $GB = \langle a, a, \neg a, b \rangle$. We get that both (a,b,c) and $(a,b,\neg c)$ beat any of the other six candidates by a majority of voters, however neither of both beats the other one by a majority of voters, therefore there is no Condorcet winner. Intuitively speaking, we would like not to distinguish $\vec{x}$ and $\vec{y}$ and to have them both as winners. For this purpose, we now define *weak Condorcet winners*.

Let us first define the equivalence relation $\sim$ on $\mathcal{X}$ by $\vec{x} \sim \vec{y}$ if and only if $\vec{x} \sim_i \vec{y}$ for every voter i (unanimous indifference between $\vec{x}$ and $\vec{y}$). A *weak Condorcet winner* (WCW) for a profile P is a candidate $\vec{x}$ such that for any other candidate $\vec{y}$, either there are strictly more agents preferring $\vec{x}$ to $\vec{y}$, or there is an unanimous indifference between $\vec{x}$ and $\vec{y}$. An equivalent definition consists in first quotienting $\mathcal{X}$ by $\sim$, and then looking for Condorcet winners in $\mathcal{X}_{/\sim}$; a candidate $\vec{x} \in \mathcal{X}$ is a WCW if and only if its equivalence class for $\sim$ is *the* CW in $\mathcal{X}_{/\sim}$.

Obviously, a CW is a WCW. Noticeably, there are profiles for which there are several WCWs (in which case there is no CW), profiles for which there is a single WCW (which is then the CW) and profiles for which there is no WCW.

When P is represented explicitly, the existence of a CW or of a WCW can be determined in quadratic time. This is much more complex when preference are represented in a succint language:

Proposition 4 *Let R be a language for which* COMPARISON *is polynomial.*

1. *determining whether $\vec{x}$ is a Condorcet winner (resp. a weak Condorcet winner) for $\mathcal{B}$ w.r.t. R is in* coNP. *It is* coNP*-complete for $R = R_{brute}$ (and a fortiori for more refined languages).*
2. *determining whether there exists a Condorcet winner (resp. a weak Condorcet winner) for $\mathcal{B}$ w.r.t. R is in Σ_2^p. For $R = R_{brute}$, the existence of a Condorcet winner is in Θ_2^p,* NP*-hard and* coNP-

[10] In particular, when $\geq$ is a complete order, $\vec{x}$ is a Condorcet winner if and only if it is preferred to any other candidate by a majority of voters.

hard)[11].

Sketch of proof: all membership proofs are easy, once noticed that for any two candidates, $\vec{x}$, $\vec{y}$, it can be decided in polynomial time whether $\vec{x} >_i \vec{y}$ for every voter i, and therefore it can be decided in polynomial time whether $|\{i|\vec{x} >_i \vec{y}\}| > |\{i|\vec{y} >_i \vec{x}\}|$, and similarly, $\vec{x} \sim \vec{y}$ can also be decided in polynomial time. For $R = R_{brute}$, we get that $\vec{x}$ is a WCW for $\mathcal{B} = \{G_1, ..., G_n\}$ if and only if $\vec{x}$ satisfies a maximal number of G_i's and $\vec{x}$ is a CW if furthermore $\vec{x}$ is the *unique* candidate maximizing the number of G_i's satisfied. This leads to three conclusions: (a) when $R = R_{brute}$, there is always a WCW, (b) checking that $\vec{x}$ is the CW is in coNP, and (c) the existence of a CW is in Θ_2^p[12].

Since the existence of a CW is not guaranteed, several voting rules have been proposed in social choice theory, which guarantee that each time there exists a CW, it is elected. These rules are said *Condorcet-consistent*. Considering now weak Condorcet winners, a Condorcet-consistent rule elects the set of WCWs when the latter is not empty.

Each Condorcet-consistent rule is thus characterized by its results in the cases where there are no WCWs.

Two simple Condorcet-consistent rules have received a particular interest, namely the *Copeland* and the *Simpson* rules. The *Copeland rule* requires to count candidates and, as said above, it is not well-suited to combinatorial vote; the Simpson rule is more interesting.

4.3.2 The Simpson rule

Let P be a profile, $\vec{x}$ and $\vec{y}$ two candidates. $N(\vec{x}, \vec{y})$ is defined as the number of voters who prefer $\vec{x}$ to $\vec{y}$, *i.e.*, $N(\vec{x}, \vec{y}) = |\{i|\vec{x} >_i \vec{y}\}|$. The *Simpson score* $S(x)$ is defined by $S(\vec{x}) = \min_{\vec{y} \neq \vec{x}} N(\vec{x}, \vec{y}) - N(\vec{y}, \vec{x})$, and $Select_{Simpson}(P)$ is the set of candidates maximizing $S(\vec{x})$ (the Simpson winners). It is easy to check that when there exists at most one $\vec{x}$ with a strictly positive score, and when such a candidate exists, it is the CW: the Simpson rule is Condorcet-consistent.

Proposition 5 *Let R be a language for which comparison is polynomial.*

1. AMONG-WINNERS$_{Simpson,R}$ *is in* Θ_2^p.
2. ELECT-SAT$_{Simpson,R}$ *in* Σ_2^p.
3. *For $R = R_{brute}$:* AMONG-WINNERS$_{Simpson,R_{brute}}$ *is* coNP*-complete and* ELECT-SAT$_{Simpson,R_{brute}}$ *is* Θ_2^p*-complete.*

Sketch of proof: the first point comes easily from the fact that $-n \leq S(\vec{x}) \leq n$ (n being the number of voters), making possible a binary search over a set of $2.n$ possible values. The second point is then derived easily from the first one. For the last point, ELECT-SAT-hardness is obtained by a straightforward reduction from cardinality-maximizing base revision [31].

Example 1 (continued).

On the above table we give the values of $N(\vec{x}, \vec{y})$ ($\vec{x}$ in column and $\vec{y}$ in row) and then the Simpson scores. We represent candidates by vectors of digits: (a, b, c) by 111, $(a, b, \neg c)$ by 110 and so on.

$\downarrow \vec{y}\ \vec{x} \rightarrow$	111	110	101	100	011	010	001	000
111	/	0	0	0	0	−1	0	0
110	0	/	0	0	0	−1	1	0
101	0	0	/	0	0	−1	1	0
100	0	0	0	/	−1	−1	0	−1
011	0	0	0	1	/	−2	−1	−2
010	1	1	1	1	2	/	0	0
001	0	−1	−1	0	1	0	/	−1
000	0	0	0	1	2	0	1	/
$S(\vec{x}) \rightarrow$	0	−1	−1	0	−1	−2	−1	−2

There are two Simpson winners, namely (a, b, c) and $(a, \neg b, \neg c)$. None of them is a weak Condorcet winner.

4.4 Discussion

The conclusions are not simple, because several parameters have to be taken into account. From a strictly computational point of view, aggregative rules are probably easier to implement than scoring rules or Condorcet-consistent rules, but they require a numerical input which is often hard to obtain, and they are thus considered little efficient from a cognitive point of view. Note that our complexity results for the Simpson rule and for the plurality rule coincide[13], namely, its complexity is not above the second level of the polynomial hierarchy. Therefore it seems that the Simpson rule, considered more acceptable than the plurality rule [29], could be considered as an acceptable trade-off.

[11] and the existence of a weak Condorcet winner is trivial (see the sketch of proof).

[12] Note that I could not manage to prove Θ_2^p-hardness, which nevertheless I strongly believe to hold. Similarly, I strongly believe that when R varies, then the upper bound, i.e., Σ_2^p-completeness, is reached for simple languages such as R_{card}, but again I could not find a proof.

[13] This does not mean, however, that the complexity of both rules would coincide for all R.

5 Future directions

5.1 Electronic vote

A first possible application of combinatorial vote is *electronic vote* concerning decisions about several dependent variables, that have to be taken in small organizations (small companies, laboratories, recruiting committees etc.). When the set of candidates has little or no combinatorial structure (e.g., choose a person to be recruited on a position), or when the variables are independent (or almost) from each other, preferences can be aggregated manually, but this is not the case as soon as the former has a strong combinatorial structure[14].

In this paper we only briefly mentioned the key issue of automated *preference elicitation* (i.e., how to interact with a voter so as to obtain her preference relation). This issue, which has received some attention in the last years, is a necessary upstream task for combinatorial vote and relationships between both problems should be studied further. As well, identifying preferential independences between some variables for a given voter (see [2]) is extremely relevant in this context.

5.2 Partial vote and negociation

In some situations, it is not compulsory to assign *all* variables at once: one may try to localize conflicts and look for a partial decision (an assignment of a subset of the variables of interest) that is both as consensual as possible and as complete as possible; this decision can then be communicated to the voters, who can then update their preferences about the remaining variables (possibly after a negociation phase which is out of our concern), and the process can then be iterated until all variables have been assigned. This process requires that voting rules for combinatorial vote be adapted; this is an issue for further work.

Other less obvious possible applications of combinatorial vote are fair division problems (see [27]) or combinatorial auctions. This is left for further research.

[14] This can be seen for instance on the following real-world problem, concerning a decision to be taken by a recruiting committee: when not a single, but k individuals (out of n) can be recruited, the space of possible decisions has a combinatorial structure: a "candidate" is no longer an individual but a set of k individuals. The problem can be solved manually only if the dependencies between individuals are ignored, which means that the voters cannot express *correlations* between individuals, as for instance: "In the first place I prefer recruiting A, then B, then C, but since A and B work on similar subjects whereas C works on a complementary subject, the joint recruiting of A and C, or even of B and C, suits me better than the joint recruting of A and B." Allowing for the expression of such preference needs a sophisticated language for preference representation such as those studied in this article, and the aggregation of such preference is clearly a combinatorial vote problem.

Acknowledgements: I would like to thank Sébastien Konieczny for a careful reading of a preliminary version of the paper, as well as Thomas Eiter and Pierre Marquis for discussions about Θ_2^p-hardness.

References

[1] F. Bacchus and A. Grove. Graphical models for preference and utility. *Proceedings of the 11th Conf. on Uncertainty in Artificial Intelligence (UAI'95)*, 3-10.

[2] F. Bacchus and A. Grove. Utility independence in a qualitative decision theory. *Proc. of KR'96*, 542-552.

[3] S. Benferhat, C. Cayrol, D. Dubois, J. Lang and H. Prade. Inconsistency management and prioritized syntax-based entailment. *Proceedings of IJCAI'93*, 640-647.

[4] S. Benferhat, D. Dubois, S. Kaci and H. Prade. Bipolar representation and fusion of preferences in the possibilistic logic framework. *Proceedings of KR-2002* (these Proceedings).

[5] C. Boutilier. Towards a logic of qualitative decision theory. *Proceedings of KR'94*, p. 75-86, 1994.

[6] C. Boutilier, R. Brafman, H. Hoos and D. Poole. Reasoning with conditional *ceteris paribus* statements. *Proceedings of the 15th Conf. on Uncertainty in Artificial Intelligence (UAI'99)*, 71-80.

[7] C. Boutilier, F. Bacchus and R. Brafman. UCP-networks: a directed graphical representation of conditional utilities. *Proceedings of UAI-2001*, 56-64.

[8] C. Boutilier and H. Hoos. Bidding languages for combinatorial auctions. *Proceedings if IJCAI'01*, 1211-1217.

[9] T. Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69, 161-204, 1994.

[10] J.-Y. Cai, T. Gundermann, J. Hartmanis, L.A. Hemachandra, V. Sewelson, K. W. Wagner and

G. Wechsung. The Boolean hierarchy I: structural properties. *SIAM Journ. on Computing*, 17 (6), 1232-1252, 1988.

[11] M. Cadoli, F. Donini, P. Liberatore and M. Schaerf. Comparing space efficiency of propositional knowledge representation formalisms. *Proc. of KR'96*, 364-373.

[12] C. Domshlak, R.I. Brafman and S. Eyal Shimony. Preference-Based Configuration of Web Page Content. *Proc. of IJCAI'2001.*

[13] D. Dubois, H. Fargier and P. Perny. On the limits of ordinality in decision making. In *Proceedings of KR-02* (these Proceedings).

[14] J. Doyle and M. P. Wellman. Preferential semantics for goals. Proceedings of *AAAI'91*, p. 698-703, 1991.

[15] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactuals. *Artificial Intelligence* 57, 227-270, 1992.

[16] T. Eiter and T. Lukasiewicz. Complexity results for default reasoning from conditional knowledge bases. *Proceedings of KR-2000*, 62-73.

[17] N. Friedman and J. Halpern. On the complexity of conditional logics. *Proceedings of KR'94*, 202-213.

[18] H. Geffner. *Default Reasoning: Causal and Conditional Theories.* MIT Press, 1992.

[19] S. Konieczny, J. Lang et P. Marquis. Distance-based merging: a general framework and some complexity results. In *Proceedings of KR-02* (these Proceedings).

[20] S. Konieczny and R. Pino-Pérez, On the logic of merging. *Proceedings of KR'98*, 488-498, 1998.

[21] C. Lafage, J. Lang. Logical representation of preferences for group decision making. *Proceedings of KR'2000*, 457-468, 2000.

[22] C. Lafage and J. Lang. Propositional distances and preference representation. *Proceedings of ECSQARU-2001, LNAI 2143*, Springer-Verlag, 48-59, 2001.

[23] P. La Mura and Y. Shoham. Expected utility networks. *Proceedings of UAI'99*, 366-373, 1999.

[24] J. Lang. Conditional desires and utilities - an alternative approach to qualitative decision theory. *Proceedings of ECAI'96*, 318-322, 1996.

[25] J. Lang and P. Marquis. Complexity results for independence and definability in propositional logic, *Proceedings of KR'98*, 356-367.

[26] J. Lang, L. van der Torre and E. Weydert. Utilitarian desires. To appear in *International Journal on Autonomous Agents and Multi-Agent Systems*, 2002.

[27] M. Lemaitre, G. Verfaillie, and N. Bataille. Exploiting a common property resource under a fairness constraint: a case study. *Proceedings of IJCAI'99*, 206-211.

[28] P. Liberatore and M. Schaerf. BReLS: a system for the integration of knowledge bases. *Proceedings of KR-2000*, 145-152.

[29] H. Moulin, *Axioms of Cooperative Decision Making*, Cambridge University Press, 1988.

[30] B. Nebel. Belief revision and default reasoning: syntax-based approaches. *Proceedings of KR-91*, 417-428.

[31] B. Nebel, How hard is it to revise a knowledge base? In D.M. Gabbay and P. Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, vol. 3, 77-145, Kluwer Academics.

[32] C. H. Papadimitriou, *Computational complexity*, Addison-Wesley Publishing Company, 1994.

[33] J. Pearl. System Z: a natural ordering of defaults with tractable applications to default reasoning. *Proceedings of TARK'90*, 121-135.

[34] T. Sandholm. An Algorithm for Optimal Winner Determination in Combinatorial Auctions, *Proc. of IJCAI'99*, 542-547.

[35] T. Schiex, H. Fargier and G. Verfaillie. Valued constraint satisfaction problems: hard and easy problems, *Proceedings of IJCAI'95*, 631-637, 1995.

[36] S.-W. Tan and J. Pearl, Specification and evaluation of preferences for planning under uncertainty, *Proc. KR'94.*

[37] R. H. Thomason. Desires and defaults: a framework for planning with inferred goals. *Proceedings of KR-2000*, 702-713.

[38] L. van der Torre and E. Weydert. Parameters for utilitarian desires in a qualitative decision theory, *Applied Intelligence*, 14 (3), 285-302, 2001.

[39] M. Wooldridge and S. Parsons. Languages for negociation. *Proceedings of ECAI-2000*, 393-397.

Reasoning about Action I

A transition function based characterization of actions with delayed and continuous effects

Chitta Baral
Computer Science and Engineering
Arizona State University
Tempe, AZ 85287, USA
chitta@asu.edu

Tran Cao Son
Computer Science Department
New Mexico State University
PO Box 30001, MSC CS
Las Cruces, NM 88003, USA
tson@cs.nmsu.edu

Le-Chi Tuan
Computer Science and Engineering
Arizona State University
Tempe, AZ 85287, USA
lctuan@asu.edu

Abstract

In this paper we present a transition function based characterization of actions in a realistic environment. Our language allows for the specification of actions with duration, continuous effects, delayed effects, dependency on non-sharable resources, and accounts for parallel and overlapping execution of actions. One of the main contribution of our paper is a new definition of state in such an environment. Our notion of state encodes not only the fluent values but also obligations due to delayed effect of actions that were executed recently. This allows us to define a Markovian transition function. Although there have been earlier attempts at developing action languages with similar features, none of them present a transition function based characterization.

1 INTRODUCTION AND MOTIVATION

In this paper we follow the approach of high level action description languages with a transition function based semantics (as first done in [11]) to characterize more realistic actions. In particular, we consider: (i) actions that have continuous effects such as the action of driving that changes the fluent recording the distance traveled, (ii) actions with fixed and variable durations, (iii) actions that use non-sharable resources such as machines, (iv) overlapping execution of actions that contribute to the change of a resource, and (v) actions with delayed effects. Although some of these features have been earlier considered in some action description languages, none of them present a transition function based semantics.

By a transition function based semantics we mean *defining what a state is and transition between states due to actions.* If we consider a state to be a snapshot of the world at a particular time, then how do we express this snapshot when actions have durations? It is not enough to use the traditional notion of a state that encodes the value of fluents and resources as it misses out the actions under execution. For example, if we go to a factory and try to express the snapshot of the factory at 12:35 PM, then it is not enough to just say the fluent values, but also important to express which actions are under execution and for how long. Only then we can correctly predict the value of fluents at a future time by only considering the current 'state'. In other words a richer notion of state is necessary to define a Markovian transition function.

The traditional transition functions define a function $\Phi(s, a)$, which gives the state that would be reached if action a was executed in state s. Since we now consider actions with durations, and allow overlapping of actions our goal is to define a function $\Phi(s, \{(a_1, t_1), \ldots, (a_n, t_n)\}, t)$, where $t_1 \leq t_2 \ldots \leq t_n \leq t$, which expresses the state of the world after t units of time (from the time t_0 corresponding to s) assuming that the actions $a_1, \ldots, a_n$ were started at relative (to t_0) times $t_1, \ldots, t_n$ respectively.

Such a transition function is important in planning and other kinds of reasoning about actions and is absent in earlier attempts [7, 15, 20, 21] of characterizing actions with such realistic features. Such a transition function based characterization is also important in light of the upcoming planning contest at AIPS'02 where planners that allow

such actions are expected to compete. In Section 3 we will give a slightly more detailed comparison of our approach with earlier reasoning about action attempts [7, 15, 20, 21], and the semantics of the languages PDDL2.1 [9] and PDDL++ [10] that have been recently proposed for use in the AIPS'02 planning contests and beyond.

2 THE LANGUAGE $\mathcal{ADC}$

We will call our language $\mathcal{ADC}$ as our language is for $\mathcal{A}$ctions with $\mathcal{D}$elayed and $\mathcal{C}$ontinuous effects.

2.1 SYNTAX OF THE DOMAIN DESCRIPTION PART

The alphabet of a domain description in $\mathcal{ADC}$ consists of a set of action names A, a set of fluent names F, and a set of process names P. Each fluent $f \in$ F has a domain $dom(f)$ associated with it that prescribes what value f can take. Each domain $dom(f)$ is also associated with a set of binary relations over it. For example, if $dom(f)$ is the set of integers then the set of binary relations over $dom(f)$ could contain the standard comparison relations $<, >, \leq, \geq, =, \neq$. Each process name is associated with a fluent definition or a update expression which will be defined precisely later on.

An *evaluable expression* is constructed from functions and fluents in the same way an arithmetic expression is constructed in C, C++ etc. Variables in an evaluable expression are fluents in F. Given the values of fluents, the value of an evaluable expression will be computed in the standard way. For example, if f is a real-valued fluent then $3 * (f + 2)^2$ is an evaluable expression and if the value of f is 2 then the value of $3 * (f + 2)^2$ is 48. The domain of the value of an evaluable expression is called the type of that expression.

An *atom* is of the form ϕ_1 **op** ϕ_2, where ϕ_1 and ϕ_2 are evaluable expressions of the same type and **op** is a binary relation associated to the type of ϕ_1 (or ϕ_2). Atoms of the form $\phi = True$ and $\phi = False$ can be simply written as ϕ and $\neg\phi$ respectively.

In $\mathcal{ADC}$, actions can have delayed and continuous effects. This means that effects of actions might not happen immediately or can last over a period of time and therefore are associated with time intervals of the form $[t_1, t_2]$ where t_1, t_2 are real numbers, $0 \leq t_1 \leq t_2$. Actions can also initiate (or terminate) a process whose duration is unknown at the time it starts. For example, the action of turning on the water tap of the bath tub will initiate the process of water flowing into the bath tub. The process will be terminated when the tap is turned off. On the other hand, flipping the switch of a lamp changes its position (on to off and off to on) and takes a constant amount of time. To express this, the various propositions in $\mathcal{ADC}$ are of the following form

$$\textbf{executable } a \textbf{ if } c_1, \ldots, c_k \tag{2.1}$$
$$a \textbf{ needs } r_1, \ldots, r_m \tag{2.2}$$
$$a \textbf{ causes } f = valf(f, f_1, \ldots, f_n, t) \textbf{ from } t_1 \textbf{ to } t_2 \tag{2.3}$$
$$a \textbf{ contributes } valf(f, f_1, \ldots, f_n, t) \textbf{ to } f \textbf{ from } t_1 \textbf{ to } t_2 \tag{2.4}$$
$$a \textbf{ initiates } p \textbf{ from } t_s \tag{2.5}$$
$$a \textbf{ terminates } p \textbf{ at } t_s \tag{2.6}$$
$$p \textbf{ is\_associated\_with } f = valf(f, f_1, \ldots, f_n, t) \tag{2.7}$$
$$p \textbf{ is\_associated\_with } f \leftarrow valf(f, f_1, \ldots, f_n, t) \tag{2.8}$$

where

- a is an action name, the f's are fluents, the c's are atoms,
- the r's are atoms of the specific form $f = \phi$ where f is a numeric-type fluent and ϕ is an evaluable expression,
- t_s, t_1, t_2 $(t_1 \leq t_2)$ are non-negative real numbers, representing time units relative to the time point where a's execution is started,
- $valf(f, f_1, \ldots, f_n, t)$ is a function that takes the value of the fluents $f, f_1, \ldots, f_n$ when a started its execution and the elapsed time t in $[0, t_2 - t_1]$ and returns a value from $dom(f) \cup \{undefined\}$, and
- p is a process name and (2.7) states that p is associated with the *fluent definition* "$f = valf(f, f_1, \ldots, f_n, t)$" whereas (2.8) says that it is associated with the *update expression* "$f \leftarrow valf(f, f_1, \ldots, f_n, t)$", where f and $valf(f, f_1, \ldots, f_n, t)$ have the same meaning as in the above item.

Intuitively, the above propositions describe the effects of actions and their executability conditions. Propositions of the form (2.1)-(2.2) state the conditions under which a is executable. Here, (2.1), called an *executability condition*, represents the requirements on certain fluents (or resources) that a does not need exclusively to start its execution and (2.2), called a *resource condition*, characterizes the conditions on fluents that a needs exclusively to start its execution. For example, a postal worker

needs his car for mail delivery s but he must be in the car for the action to be executable.

Propositions of the form (2.3) and (2.4) describe the different ways in which an action can affect the value of a fluent. In (2.3), a causes the value of f to change according to the function $valf(f, f_1, \ldots, f_n, t)$ during the interval $[t_1, t_2]$. On the other hand, a in (2.4) affects the value of f at t_1 by contributing an increase specified by $valf(f, f_1, \ldots, f_n, t)$ during the interval $[t_1, t_2]$. We require that in (2.3) if $t_1 = 0$ then $valf(f, f_1, \ldots, f_n, 0) = f$ and in (2.4) if $t_1 = 0$ then $valf(f, f_1, \ldots, f_n, 0) = 0$.

The purpose of the propositions (2.5) and (2.6) is to encode continuous effect of actions that do not have a predefined duration. In that case a proposition of the form (2.5) is used to indicate which actions can initiate the effect (a process) and a proposition of the form (2.6) is used to indicate which actions can terminate the process. Note that unlike in some earlier proposal [20, 21, 10] we do not designate these actions as start and end actions. In our framework the same process can be initiated and terminated by different actions. For example, the termination of the filling of a bathtub can be caused by turning off the bath faucet and also by turning off the main water switch.

We now give some examples to illustrate the kind of domain description that can be expressed in $\mathcal{ADC}$.

Example 1 Consider the action of driving a car for 10 units of time with a velocity v. We represent this action *of fixed duration* by $drive_{0,10}(v)$. Now, let *gas_in_tank* denote the amount of gasoline available in the tank.

The executability condition saying that the action $drive_{0,10}(v)$ is executable if $gas\_in\_tank \geq 20$ can be expressed in $\mathcal{ADC}$ as:

executable $drive_{0,10}(v)$ **if** $gas\_in\_tank \geq 20$.

The effect of the action $drive_{0,10}(v)$ on the fluent *loc* (encoding how far from the initial position the car is) is expressed as:

$drive_{0,10}(v)$ **causes** $locn = locn + v * t$
from 0 **to** 10.

We can record the information that the action $drive_{0,10}(v)$ is under execution by the following propositions:

$drive_{0,10}(v)$ **causes** *driving* **from** 0 **to** 10

and

$drive_{0,10}(v)$ **causes** $\neg$*driving* **from** 10 **to** 10

where *driving* is a Boolean-fluent.

We can express that driving at a velocity v consumes $c(v)$ unit of gasoline per unit time as

$drive_{0,10}(v)$ **contributes** $-c(v) * t$
to *gas_in_tank* **from** 0 **to** 10.

Intuitively, if $gas\_in\_tank = 20$ and $c(3) = 1.5$ at time 0, then the level of gasoline in the tank at the time $0, 1, \ldots, 10$ will be as follows, assuming that $drive_{0,10}(3)$ is executed at the time 0:

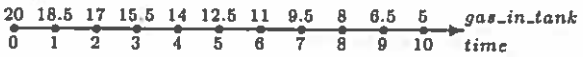


Figure 1: Gasoline in tank and time – $drive_{0,10}(3)$ at 0

The proposition

$fill\_gas_{0,10}$ **contributes** $2 * t$ **to** *gas_in_tank*
from 0 **to** 10

expresses the contribution of the action $fill\_gas_{0,10}$ to the value of the *gas_in_tank*. Intuitively, it says that the action of filling gas for 10 units of time will increase the amount of gasoline, by the amount specified by the function $2 * t$. Let us assume that while driving we can refill. The following diagram shows the value of *gas_in_tank* at different times, assuming that the action $drive_{0,10}(3)$ starts its execution at the time moment 0 and the action $fill\_gas_{0,10}$ starts its execution at the time moment 1:

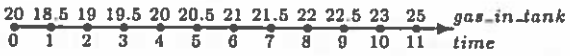


Figure 2: Gasoline in tank and time – $drive_{0,10}(3)$ at 0 and $fill\_gas_{0,10}$ at 1

Example 2 Let us now consider the action of driving when the duration is not fixed beforehand. In that case we have the following propositions:

start_drive(3) **initiates** p_1 **from** 0
start_drive(3) **initiates** p_2 **from** 0
stop_drive(3) **terminates** p_1 **at** 0
stop_drive(3) **terminates** p_2 **at** 0
p_1 **is_associated_with** $locn = locn + 3 * t$
p_2 **is_associated_with** $gas\_in\_tank \leftarrow -1.5 * t$

This says that the process of increasing the fluent *loc* and the process of reducing the amount of gasoline in the tank will start at the time *start_drive*(3) is executed. These processes will

be stopped when the action *stop_drive*(3) is executed. □

A *domain description* D is a set of propositions of the form (2.1)-(2.8). For simplicity, we require that for each action a in D, there is at most one proposition of the form (2.2) whose action is a. An *action theory* is a pair (D, O) where D is a domain description and O is a set of *observations* of the form

$$\textbf{initially } f = c \tag{2.9}$$

where f is a fluent and c is a value belonging to $dom(f)$.

2.2 QUERIES

In the presence of delayed and continuous effects, we are interested not only about the values of fluents at certain time point but also about the intervals of them and whether the value satisfy some conditions or not. For example, we might be interested in knowing whether the amount of gasoline in the tank is more than 5 units or not; or whether the car is at certain location after driving for some unit of time; or a combination of both questions etc. This leads us to consider *queries* of the following form:

$$c_1[\delta_1^-, \delta_1^+], \ldots, c_k[\delta_k^-, \delta_k^+] \textbf{ after } A_1 : t_1, \ldots, A_n : t_n \tag{2.10}$$

where $c_1, \ldots, c_k$ are atoms, the δ's are time units with $0 \leq \delta_i^- \leq \delta_i^+$, $A_1, \ldots, A_n$ are set of actions, and $0 \leq t_1 < \ldots < t_n$ are time units. The intuitive reading of query (2.10) is that whether the atoms c_i are satisfied during the interval δ_i^-, δ_i^+ for $1 \leq j \leq k$ given that the A_j are executed at the moment t_j $(1 \leq j \leq n)$. All the time variables t's are relative to the current situation and the δ's are relative to t_n.

For the action theory in Example 1, let us assume that initially $gas\_in\_tank = 25$, $loc = 0$, and driving with the speed of 3 units of distance per unit of time will consume 1.5 units of gasoline per unit of time. The query

$$(gas\_in\_tank \geq 5)[5, 6], (loc = 30)[10, 10] \textbf{ after } \{drive_{0,10}(3)\} : 0$$

asks whether the amount of gasoline is greater than or equal to 5 (units of gasoline) during the interval $[5, 6]$ after driving with the speed of 3 for 5 units of time and whether the location of the car is 30 at the time moment 10.

A more complicated query involving the driving action and the *fill_gas* action is

$$(gas\_in\_tank = 20)[15, 15] \textbf{ after } \{drive_{0,10}(3)\} : 0, \{fill\_gas_{0,10}\} : 10$$

which asks whether the amount of gasoline in the tank is 20 at the time moment 15 after driving and then refilling it at the time moment 10.

2.3 SEMANTICS

We will now present a transition function based semantics for action theories in $\mathcal{ADC}$ that facilitates the answering of queries of the form (2.10) given an action theory (D, O). As it is customary in approaches to reasoning about action and change using high-level action description languages we will define the notion of a state of the world and characterize the transitions between states.

2.3.1 States: Interpretation Of Fluents And Obligations

Given a domain description D, an *interpretation* I of D assigns each fluent $f \in \mathbf{F}$ a value $v \in dom(f) \cup \{undefined\}$, denoted by $f = v$. Value of an evaluable expression x with respect to I, denoted by $I(x)$, is defined inductively over their structure in the usual way[1].

An interpretation I satisfies an atom x_1 **op** x_2 if x_1 and x_2 are defined with respect to I and belong to the same domain, say $dom(f)$ for some fluent f, **op** is a binary relation associated to $dom(f)$ and $I(x_1)$ **op** $I(x_2)$ holds. When I satisfies an atom a, we write $I \models a$. I satisfies a set of atoms $\{a_1, \ldots, a_n\}$, written as $I \models \{a_1, \ldots, a_n\}$, if $I \models a_i$ for every i, $1 \leq i \leq n$.

In transition function based approaches to reasoning about action and change (for example, in [2, 3, 4, 11, 13, 12, 14, 17, 18, 23, 25]), a state of the world is represented by a set of fluents. The semantics of an action theory is specified by a transition function that maps pairs of states and actions into states. In the presence of time and delayed effects, the effects of an action might not happen immediately or might last over a time interval. This stipulates us to represent the current state of the world by a *snapshot of the world recording the fluent values and the future obligations due to*

[1] Because a function can return *undefined* as an answer, *undefined* could be a possible value of $I(x)$. In this case, we say that x is undefined with respect to I.

delayed effects of recent actions and actions that are under execution. We elaborate more on this point below.

Consider the action $drive_{0,10}(3)$ from Example 1. Suppose that we start $drive_{0,10}(3)$ at the time moment 0. We expect that the fluent *loc* will change its value from $loc_0 = 0$ to 30 during the interval $[0, 10]$ (according to the equation $3 * t$) and the fluent *gas_in_tank* will decrease its value by 1.5 unit every unit of time. That is, at the time moment 0, when $drive_{0,10}(3)$ starts its execution, we would express the obligation due to this action on *loc* and *gas_in_tank* by equations of the form

$$loc = loc_0 + 3 * t \text{ and } gas\_in\_tank -= 1.5 * t$$

where $t \in [0, 10]$. At a later moment, say 4 (i.e, after 4 units of time), although these equations do not change the constraint on the time variable does, and they would become

$$loc = loc_0 + 3 * t \text{ and } gas\_in\_tank -= 1.5 * t$$

where $t \in [4, 10]$. Equivalently, we can write

$$loc = loc_0 + 3*(t+4) \text{ and } gas\_in\_tank -= 1.5*(t+4)$$

where $t \in [0, 6]$. These equations differ from the previous equations in that t, the time variable, refers to the time elapsed from *now* instead of the elapsed time from the beginning of the interval. However, to represent the correct value of the fluent, the time between the beginning of the interval and *now* – which is 4 in the last two equations – must be also recorded. We represent this by the interval $[-4, 6]$ where the negative part of the interval represents the elapsed time since the beginning of the obligation while the positive part says how long more the obligation lasts. We note that $[-4, 6]$ can be obtained from $[0, 10]$ by shifting it to the left by 4 units.

The above discussion suggests that we represent a state by a pair $\langle I, Q \rangle$, hereafter called a *snapshot*, where I is an interpretation and Q is a set of future effects (or obligations, for short). To distinguish an obligation caused by actions in (2.3)-(2.4) from an obligation initiated by actions in (2.5)-(2.6), we attach to each obligation a name. We use T for the former and the process name for the latter. Further, we will use $[t_s, \infty)$ to represent the interval of an obligation generated by a process to indicate that we do not know when such an obligation will terminate[2]. Obligations caused by an action in (2.3)-(2.5) do change the world but obligations caused by (2.6) do not. Instead, they terminate obligations caused by (2.5). For this reason, we only add to Q obligations caused by actions in (2.3)-(2.5). Whenever an action in (2.6) is executed we will update Q by removing a matching obligation, if any. An obligation is one of the following form

$$(\mathsf{T}, f = valf(v_f, v_{f_1}, \ldots, v_{f_n}, t), t_1, t_2) \quad (2.11)$$
$$(\mathsf{T}, f \leftarrow valf(v_f, v_{f_1}, \ldots, v_{f_n}, t), t_1, t_2) \quad (2.12)$$
$$(p, f = valf(v_f, v_{f_1}, \ldots, v_{f_n}, t), t_1, \infty) \quad (2.13)$$
$$(p, f \leftarrow valf(v_f, v_{f_1}, \ldots, v_{f_n}, t), t_1, \infty) \quad (2.14)$$

where

- f is a fluent,
- t_1, t_2 are two real numbers with $t_1 \leq t_2$, and
- $valf(v_f, v_{f_1}, \ldots, v_{f_n}, t)$ is a function with $v_f, v_{f_1}, \ldots, v_{f_n}$ are the values of the fluents $f, f_1, \ldots, f_n$ at the time the obligation is added to Q.

In the following, we will refer to obligations of the form (2.11) or (2.13) as *type-1* obligations. The other obligations are called *type-2* obligations.

For convenience, we refer to the elements of an obligation e by $e.name$, $e.f$, $e.t_1$, and $e.t_2$ and call f, t_1, and t_2 as the fluent, lower, and upper bound of e, respectively. We sometimes use $e.[t_1, t_2]$ or $e.[t_1, \infty)$ to represent the interval specified by the lower and upper bound of e. A fluent f *occurs* in an obligation e if f is the left hand side of $e.f$.

Intuitively, an obligation $e = (e.name, e.f, e.t_1, e.t_2)$ states that during the time interval $e.[t_1, t_2]$ (or $e.[t_1, \infty)$), the value of a fluent f (the left hand side of $e.f$) will be determined (or increased by an amount) as specified by the function on the right hand side of $e.f$. As time progresses, $e.t_1$ and $e.t_2$ will be updated accordingly. Furthermore, since an action might have delayed effects, we distinguish obligations whose interval contains 0 ($0 \in e.[t_1, t_2]$) from others and call them *active obligations*. Obviously, to determine what is the current value of a fluent, we only need to consider the active obligations.

Example 3 Let us consider the case where the action $drive_{0,10}(3)$ in Example 1 is executed at the time 0 with the initial value of *loc* equal to 0. Then,

- The type-1 obligation $(\mathsf{T}, loc = 3 * t, 0, 10)$ is added to Q. After one unit of time, this

[2] Adding the name to the obligation or ∞ to the interval gives us a uniform representation of obligations.

obligation will be changed to $(\top, loc = 3 * t, -1, 9)$.
- The type-2 obligation $(\top, gas\_in\_tank \leftarrow -1.5t, 0, 10)$ is added to Q. After one unit of time, it will be changed to $(\top, gas\_in\_tank \leftarrow -1.5t, -1, 9)$.

On the other hand, if *start_drive*(3) (Example 2) is executed at the time 0, then

- The type-1 obligation $(p_1, loc = 3 * t, 0, \infty)$ is added to Q. After one unit of time, the obligation will be changed to $(p_1, loc = 3 * t, -1, \infty)$.
- The type-2 obligation $(p_2, gas\_in\_tank \leftarrow -1.5t, 0, \infty)$ is added to Q. After one unit of time, it will be changed to $(p_2, gas\_in\_tank \leftarrow -1.5t, -1, \infty)$.

If *stop_drive*(3) is executed at the moment 1, then its effect is to remove the obligations named p_1 and p_2 from Q. Therefore, Q will no longer contain the obligations $(p_1, loc = 3 * t, -1, \infty)$ and $(p_2, gas\_in\_tank \leftarrow -1.5t, -1, \infty)$

Now that we have defined what a state is, we need to discuss when a snapshot $\langle I, Q\rangle$ represents a valid state of the world. It is easy to see that the obligations of a snapshot dictate the values of fluents in the future. Further, to determine the value of a fluent, say f, at a time moment δ, we need to consider all the obligations that will assign a value for f (or contribute some increase for f) at δ. We call these obligations *active* at δ and characterize them in the next definition.

Definition 1 Let $\langle I, Q\rangle$ be a snapshot and $\delta \geq 0$. An obligation $e = (e.name, e.f, e.t_1, e.t_2)$ is *active at* δ if $e.t_1 \leq \delta \leq e.t_2$. □

Because actions in $\mathcal{ADC}$ can be executed in parallel and different actions might assign different value to a fluent f at some future moment of time, there might be situations in which two type-1 obligations assign different value to a fluent f simultaneously. The following picture depicts such a situation:

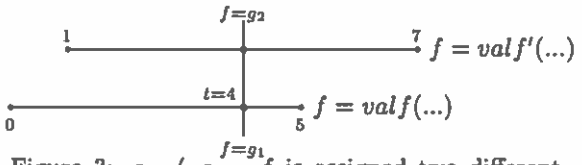


Figure 3: $g_1 \neq g_2$ – f is assigned two different value at time 4

We consider this a *conflict situation* and call a snapshot that causes a conflict situation an *inconsistent snapshot*. We formulate precisely this notion in the next definition.

Definition 2 A snapshot $\langle I, Q\rangle$ is said to be *inconsistent* if one of the following conditions is satisfied:

1. there exists a time moment $\delta \geq 0$ and two type-1 obligations e and e', both are active at δ, with $e.f$ is "$f = valf(v_f, v_{f_1}, \ldots, v_{f_n}, t)$" and $e'.f$ is "$f = valf'(v_f, v_{g_1}, \ldots, v_{g_m}, t)$" and there exists a δ' such that $0 \leq \delta' < \delta$ and
   - both e and e' are active at δ', and
   - $valf(v_f, v_{f_1}, \ldots, v_{f_n}, \delta - e.t_1) \neq valf'(v_f, v_{g_1}, \ldots, v_{g_m}, \delta - e'.t_1)$.
2. there exists a time moment $\delta \geq 0$, a fluent f, a type-1 obligation e, and a type-2 obligation e' such that (i) f occurs in e and e', and (ii) both e and e' are active at δ.

We say that $\langle I, Q\rangle$ is *consistent* if it is not inconsistent. □

In the above definition, in condition 1 we require the existence of δ' that is smaller than δ and both e and e' are active at δ'. The rationality behind this decision is similar to that of allowing obligations of actions to override the current values of fluents. More precisely, consider two type-1 obligations e and e', both assign value to f. Suppose that e' happens later than e ($e.t_1 \leq e'.t_1$). Thus, at the time moment $e'.t_1$, value of f dictated by e should be considered as the value by *inertia* whereas the *new* value of f dictated by e' should be considered the value of f. This said, if $e.[t_1, t_2]$ and $e'.[t_1, t_2]$ share some points then the value of f dictated by e and e' should be the same on every point common to these intervals except the lower bound of e'[3]. Condition 2 requires that a fluent is not assigned a value by an action and updated by another at the same time.

2.3.2 Transition Function

In this section we present several definitions leading to the definition of a transition function. Since we have to go through several definitions, we start with a road map. As mentioned in the intro-

[3] Another possible way of dealing with this problem is to ignore the type-1 obligation that is currently active when a new type-1 obligation on the same fluent becomes active. This seems not natural because under this view, every possible snapshot is valid which is unlikely in most real-life domains.

duction, our objective here is to define the function $\Phi(s, \{(a_1, t_1), \ldots, (a_n, t_n)\}, t)$. We do this by defining a function $\tau(s, \delta)$ (Definition 6) that gives us the state after the passage of δ units of time when starting from the state s. (Note that this is not s. Even though no new action has started some of the obligations in s may become due because of passage of time.) The definition of $\tau(s, \delta)$ involves defining the new interpretation denoted by $I^{Q,\delta}$ (Definitions 3-4) and the new obligations denoted by $Shift(Q, \delta)$ (Definition 5).

Our next goal is to define a function $\Phi(s, A, \delta)$ (Definition 10) that defines the state reached after executing the set of actions A immediately in the state s and then waiting for δ time units. $\Phi(s, \{(a_1, t_1), \ldots, (a_n, t_n)\}, t)$ is then defined using $\Phi(s, A, \delta)$ by sorting $\{(a_1, t_1), \ldots, (a_n, t_n)\}$ into $\{(A'_1, t'_1), \ldots, (A'_k, t'_k)\}$ such that each set $A'_i \subseteq \{a_1, \ldots, a_n\}$, and for any i, if $a \in A'_i$ then $(a, t'_i) \in \{(a_1, t_1), \ldots, (a_n, t_n)\}$; and progressing through them one by one.

To define $\Phi(s, A, \delta)$ we (i) use τ, (ii) define executability of A in s (Definition 7) (iii) define the new obligations due to A denoted by $E(A, s)$ (Definition 8), and (iv) define updating of obligations because of possible removals caused by A denoted by $Update(Q, T)$ (Definition 9).

We now define τ that maps a pair of a snapshot s and a non-negative number δ into a snapshot $\tau(s, \delta)$. Let us assume that $\tau(s, \delta) = \langle I', Q' \rangle$. We will define I' by showing how to compute the value of a fluent f at δ given a consistent snapshot $\langle I, Q \rangle$. We need the following definition.

Definition 3 Let $\langle I, Q \rangle$ be a consistent snapshot, f be a fluent, and δ be a non-negative number. A type-1 obligation e_0 is called the *most recent type-1 obligation of f relative to δ* if f occurs in e_0 and $e_0.t_2 = \max\{e'.t_2 \mid f \text{ occurs in } e' \in Q, e'.t_2 < \delta\}$. □

The above definition identifies the most recent type-1 obligation relative to δ that affects the value of f. Let us call this obligation e. Intuitively, e dictates the value of f starting from $e.t_2$ to δ if no type-1 obligation that changes f is present. And, if there are contributions to f from $e.t_2$ to δ, then they should be added to its value at $e.t_2$. We will use this to define the value of f at δ in the next definition.

Definition 4 Let $\langle I, Q \rangle$ be a consistent snapshot and δ be a non-negative number. The *value of a fluent f with respect to $\langle I, Q \rangle$ at δ*, denoted by $I^{Q,\delta}(f)$, is defined as follows. First, let e_0 be a most recent type-1 obligation of f relative to δ. Let $t_i = e_0.t_2$ and $I^*(f) = valf'(v_f, v_{h_1}, \ldots, v_{h_k}, e_0.t_2 - e_0.t_1)$ if e_0 exists and $t_i = 0$ and $I^*(f) = I(f)$, otherwise. Let

$\Gamma(f) = \{\delta v \mid \delta v = valf(v_f, v_{f_1}, \ldots, v_{f_n}, \delta - e.t_1) - valf(v_f, v_{f_1}, \ldots, v_{f_n}, 0)$ where:
there exists a type-2 obligation $e \in Q$ with
(i) e is active at some δ', $t_i \leq \delta' \leq \delta$; and
(ii) $e.f =$ is "$f \leftarrow valf(v_f, v_{f_1}, \ldots, v_{f_n}, t)$"$\}$.

- For f, that does not occur in any type-1 obligation e in Q, that is active at δ, if $\Gamma(f) \neq \emptyset$ then $I^{Q,\delta}(f) = I^*(f) + \Sigma_{v \in \Gamma(f)} v$,; otherwise, $I^{Q,\delta}(f) = I^*(f)$,
- For f, that occurs in an type-1 obligation e in Q, that is active at δ, with $e.f$ is "$f = valf(v_f, v_{g_1}, \ldots, v_{g_m}, t)$" and $e.t_1 = \max\{e'.t_1 \mid f \text{ occurs in } e' \in Q, e' \text{ is active at } \delta\}$, $I^{Q,\delta}(f) = valf(v_f, v_{g_1}, \ldots, v_{g_n}, \delta - e.t_1)$.[4]

□

In the above definition, the set $\Gamma(f)$ contains all the contributions to the fluent f at δ which are specified by type-2 obligations. The value of f is then defined using the sum of all the contributions made to f (first case) or the value dictated by a type-1 obligation (second case).

Example 4 Consider the domain description of the action $drive_{0,10}(3)$ from Example 1, the interpretation $I = \{loc = 0, gas\_in\_tank = 20\}$, and the set of obligations $Q = \{(\top, loc = 3 * t, 0, 10), (\top, gas\_in\_tank \leftarrow -1.5 * t, 0, 10)\}$. For $0 \leq \delta \leq 10$, we have that $\Gamma(gas\_in\_tank) = \{-1.5\delta\}$, which gives

- $I^{Q,\delta}(loc) = 3 * t = 3\delta$ (because of the type-1 obligation in Q), and
- $I^{Q,\delta}(gas\_in\_tank) = I(gas\_in\_tank) - 1.5\delta = 20 - 1.5\delta$ (because of the type-2 obligation in Q).

[4] The consistency requirement on $\langle T, Q \rangle$ suggests that f does not occur in any type-1 obligations that are active at δ. Hence, we do not need to worry about $\Gamma(f)$ in this case. However, f might occur in more than one type-1 obligations. As we have discussed after Definition 2, the value of f at δ should be obtained from the type-1 obligation whose upper bound is greatest among them. For instance, if $drive_{0,10}(v)$ (Example 1) starts at the time 0, then the intuitive value of *driving* at the moment 10 is false which is also the value specified by the type-1 effect $(\top, driving = false, 10, 10)$ instead of the type-1 effect $(\top, driving = true, 0, 10)$.

Definition 4 tells us how to compute the interpretation of $\tau(s,\delta)$. To complete the definition of $\tau(s,\delta)$, we need to specify how its set of obligations should be computed. Obviously, we have to eliminate from Q all the obligations that are expired, i.e., their upper bound is smaller than δ. Furthermore, we have to update the obligations's interval. We call this a shifting of Q and define it as follows.

Definition 5 For a set of obligations Q and a time moment $\delta \geq 0$, a shifting of Q by δ, denoted by $Shift(Q,\delta)$, is obtained from Q by :

- Removing from Q every obligation e with $e.t_2 < \delta$,
- For every obligation e, that remains after the first step, replacing $e.t_1$ and $e.t_2$ with $e.t_1 - \delta$ and $e.t_2 - \delta$, respectively. □

We now define the *step transition function* τ, that maps pairs of snapshots and time moments into snapshots.

Definition 6 (Step Transition Function) Let $s = \langle I, Q\rangle$ be a snapshot and $\delta \geq 0$. $\tau(s,\delta)$ is defined as follows.

1. if $\langle I, Q\rangle$ is inconsistent then $\tau(s,\delta)$ is undefined;
2. otherwise, $\tau(s,\delta) = \langle I', Q'\rangle$ where $I'(f) = I^{Q,\delta}(f)$ for every $f \in \mathbf{F}$ and $Q' = Shift(Q,\delta)$

□

As $\mathcal{ADC}$ allows actions to occur in parallel, we need to characterize when a set of actions can be executed in parallel. In [4], concurrent actions have been discussed and different ways in dealing with concurrency have been proposed. The approach in [4] assumes that actions are instantaneous. Reiter modified the situation calculus to deal with time and concurrency [21]. His approach, however, requires that the action theory contains preconditions for concurrent actions which are represented by a set of simple actions. In the next definition, we characterize when a set of actions can be executed at the same time.

Definition 7 Let $s = \langle I, Q\rangle$ be a snapshot and A be a set of actions. We say that A is executable in s if,

1. every action $a \in A$ is executable in s, i.e., for every $a \in A$ there exists a executable condition **executable** a **if** $c_1, \ldots, c_k$ and $I \models c_i$ for every c_i; and
2. $I \models f \geq \Sigma_{(f,\phi)\in S} I(\phi)$, where $S = \{(f,\phi) \mid$ there exists an action $a \in A$ and $f = \phi$ is an atom belonging to the resource condition whose action is $a\}$. □

The intuition behind the above definition is that two actions can be executed in parallel only when the exclusive resources needed for both actions are available. For example, consider a simple job-shop scheduling problem with two tasks t_1 and t_2. Both t_1 and t_2 require a machine m. This requirement can be expressed in $\mathcal{ADC}$ by the two resource conditions "t_1 **needs** $m = 1$" and "t_2 **needs** $m = 1$". Clearly, t_1 and t_2 can be executed in parallel only if we have at least 2 machines (m). This is expressed by the second condition of Definition 7. We note that these two resource conditions cannot be replaced by the two executable conditions "**executable** t_1 **if** $m = 1$" and "**executable** t_2 **if** $m = 1$" because this would imply that the two tasks can be executed in parallel when there is only one machine m. Next, we define $E(a,s)$, the obligations due to execution of a in s.

Definition 8 The obligations of an action a in a snapshot $s = \langle I, Q\rangle$ denoted by $E(a,s)$ contains

- a type-1 obligation $(\top, f = valf(v_f, v_{f_1}, \ldots, v_{f_m}, t),\ t_1, t_2)$ for each proposition "a **causes** $f = valf(f, f_1, \ldots, f_m, t)$ **from** t_1 **to** t_2",
- a type-2 obligation $(\top, f \leftarrow valf(v_f, v_{f_1}, \ldots, v_{f_m}, t), t_1, t_2)$ for each proposition "a **contributes** $valf(f, f_1, \ldots, f_m, t)$ **to** f **from** t_1 **to** t_2",
- a type-1 obligation $(p, f = valf(v_f, v_{f_1}, \ldots, v_{f_m}, t), t_s, \infty)$ for each proposition "a **initiates** p **from** t_s" and "p **is_associated_with** $f = valf(f, f_1, \ldots, f_m, t)$",
- a type-2 obligation $(p, f \leftarrow valf(v_f, v_{f_1}, \ldots, v_{f_m}, t), t_s, \infty)$ for each proposition "a **initiates** p **from** t_s" and "p **is_associated_with** $valf(f, f_1, \ldots, f_m, t)$ **to** f",

where $valf(v_f, v_{f_1}, \ldots, v_{f_m}, t)$ is the function over time obtained from $valf(f, f_1, \ldots, f_m, t)$ by simultaneously replacing every occurrence of the fluents $f, f_1, \ldots, f_m$ with their values with respect to I, $v_f, v_{f_1}, \ldots, f_{f_m}$, respectively.

For a set of actions A, let $E(A,s) = \cup_{a\in A} E(a,s)$. □

Since an action a may also terminate some obligations, we define the set $T(a,s) = \{(p,t_s) \mid$

"a **terminates** p **from** t_s" belongs to D} that lists the obligations that are terminated by a. Let $T(A,s) = \cup_{a \in A} T(a,s)$. The next definition shows how to update a set of obligations given a set of terminations of the form (p,t).

Definition 9 Let Q be a set of obligations and T be a set of pairs (p,t) where p is a process name and t is a time variable. The *update* of Q with respect to T, denoted by $Update(Q,T)$, is defined as the set of obligations obtained from Q by

- Removing from Q all effects $(e.name, e.f, e.t_1, e.t_2)$ for which T contains a pair $(e.name, t)$ such that $t \leq e.t_1$,
- For each remaining e in Q with $e = (e.name, e.f, e.t_1, e.t_2)$ if T contains a pair $(e.name, t)$ such that $e.t_1 \leq t \leq e.t_2$, then replacing $e.t_2$ with t. □

We now define $\Phi(s,A,\delta)$ and $\Phi(s, \{(A_1,t_1),\ldots,(A_n,t_n)\},\delta)$.

Definition 10 (Transition Function) Let D be a domain description, A be a set of actions, $s = \langle I, Q\rangle$ be a snapshot, and $\delta \geq 0$. We define $\Phi(s,A,\delta)$ as follows.

1. if A is not executable in s or s is inconsistent, then $\Phi(s,A,\delta)$ is undefined;
2. if A is executable in s, then $\Phi(s,A,\delta) = \tau(\langle I, Update(E(A,s) \cup Q, T(A,s))\rangle, \delta)$. □

Definition 11 For a snapshot s, a sequence of action occurrences $[A_1 : t_1], \ldots, [A_n : t_n]$ such that $0 \leq t_1 < t_2 < \ldots < t_n$ and a time moment $\delta \geq t_n$, we define the sequence of snapshots $s_0, \ldots, s_n$ as follows.

- $s_0 = \Phi(s, \emptyset, t_1)$,
- for $i = 1, \ldots, n-1$, $s_i = \Phi(s_{i-1}, A_i, t_{i+1} - t_i)$, and
- $s_n = \Phi(s_{n-1}, A_n, \delta - t_n)$.

We denote s_n by $\Phi(s, \{(A_1,t_1),\ldots,(A_n,t_n)\},\delta)$. □

The following proposition follows directly from the above definition.

Proposition 1 For A's, t's, δ, and s that satisfy the conditions given in Definition 11, we have that $\Phi(s, \{(A_1,t_1),\ldots,(A_n,t_n)\},\delta) = \Phi(\Phi(s, \{(A_1,t_1),\ldots,(A_{n-1},t_{n-1})\}, t_n - t_{n-1}), A_n, \delta - t_n)$. □

The significance of this proposition is that it allows us to compute the state resulting from the execution of a sequence of action occurrences step by step; thus allowing the development of a forward planner that can exploit domain specific knowledge to improve its efficiency. Similar planners for action theories with actions of fixed duration include TLPlan [1], TALplan [8] and SHOP [19].

2.3.3 Query Entailment

We now define when a query is entailed by a domain description. We first state when a temporal atom of the form $c[t_1,t_2]$ is true in a snapshot s.

Definition 12 A snapshot s entails a temporal atom of the form $c[t_1,t_2]$ if (i) s is consistent; and (ii) for every $\delta \in [t_1,t_2]$, $\tau(s,\delta) \models c$. □

In the next definition, we define what is the initial state of a domain description.

Definition 13 (Initial State) Let (D,O) be an action theory. An *initial state* s_0 of (D,O) is a snapshot $\langle I_0, \emptyset\rangle$ of D that satisfies every observation in O, i.e. $I_0 \models f = v$ for every observation "**initially** $f = v$" in O. □

Definition 14 (Entailment) Let (D,O) be an action theory. A query

$c_1[\delta_1^-, \delta_1^+], \ldots, c_k[\delta_k^-, \delta_k^+]$
after $A_1{:}t_1, \ldots, A_n{:}t_n$

is entailed by (D,O), denoted by

$(D,O) \models c_1[\delta_1^-, \delta_1^+], \ldots, c_k[\delta_k^-, \delta_k^+]$
after $[A_1 : t_1], \ldots, [A_n : t_n]$,

if for every initial state s_0 of (D,O), s is defined and $s \models c_i[\delta_i^-, \delta_i^+]$ for every i, $1 \leq i \leq k$, where $s = \Phi(s_0, \{(A_1,t_1),\ldots,(A_n,t_n)\}, 0)$, and Φ is the transition function of D. □

We illustrate the last definition through the following proposition and its proof.

Proposition 2 Consider the action theory (D,O) where O consists of two observations

initially $loc = 0$, and
initially $gas\_in\_tank = 20$,

and D is the domain description consisting of the actions $drive_{0,10}(3)$, $fill\_gas_{10}$, $start\_drive(3)$, and $stop\_drive(3)$ and the fluents loc and gas_in_tank from Example 1. Then, $(D,O) \models (gas\_in\_tank \geq 5)[5,6]$ **after** $\{drive_{0,10}(3)\} : 0$.

Proof: (sketch) First, it is easy to see that the only initial state of (D,O) is the snapshot $s_0 = \langle I_0, \emptyset\rangle$ with $I_0(loc) = 0$ and $I_0(gas\_in\_tank) = 20$. Furthermore, $\tau(s_0, 0) = s_0$.

Since $I_0 \models gas\_in\_tank > 0$ and D does not contain a proposition of the form (2.2) whose action is $drive_{0,10}(3)$, we conclude that $drive_{0,10}(3)$ is executable in s_0. $E(drive_{0,10}(3), s_0)$ consists of a type-1 obligation $(\top, loc = 3t, 0, 10)$ and a type-2 obligation $(\top, gas\_in\_tank \leftarrow -1.5t, 0, 10)$ (see Example 3). By definition, we have that

$$\begin{aligned}
&\Phi(s_0, \{(drive_{0,10}(3), 0)\}, 0)\\
&\quad = \tau(\langle I_0, E(drive_{0,10}(3) \cup \emptyset)\rangle, 0)\\
&\quad = \tau(\langle I_0, \{(\top, loc = 3t, 0, 10),\\
&\qquad\quad (\top, gas_in_tank \leftarrow -1.5t, 0, 10)\}\rangle, 0)\\
&\quad = \langle I_0, \{(\top, loc = 3t, 0, 10),\\
&\qquad\quad (\top, gas_in_tank \leftarrow -1.5t, 0, 10)\}\rangle\\
&\quad = s_1.
\end{aligned}$$

It follows from the computation in Example 4 that $\tau(s_1, \delta) \models (gas\_in\_tank \geq 5)$ for $\delta \in [5, 6]$. Thus, $s_1 \models (gas\_in\_tank \geq 5)[5, 6]$. In other words, we have proved that $(D, O) \models (gas\_in\_tank \geq 5)[5, 6]$ **after** $\{drive_{0,10}(3)\} : 0$. □

3 RELATED WORK, CONCLUSION AND FUTURE WORK

In this paper we have presented a transition function based characterization of actions in presence of unsharable resources, with durations, and delayed and continuous effects. To make the presentation in this extended abstract simpler we have on purpose simplified the language a bit by not allowing conditional effects, and instead added a lot of explanations and examples. We would like to point out that by using delayed effects we can express actions with fixed durations as we have shown through many examples. The main contribution of our paper is the notion of state that not only has fluent interpretations but also *obligations*, and a transition function.

Although actions with durations and continuous effects were earlier considered, none of them presented a transition function based characterization. Our characterization allows us to easily map it to a model generation based planner. We describe such a planner based on logic programming in [6] and example codes can be found in http://www.public.asu.edu/~cbaral/resources/.

In terms of related work, the three main works with similar goals as ours are [20, 21], [7, 15], and [10]. In [20, 21], Reiter adds an explicit notion of time to situation calculus and uses start and stop actions to characterize processes. Among the main differences between his work and ours is that (i) his time is "actual time" while all our notions of time are relative times, (ii) he needs triggers to characterize actions with fixed durations, (iii) he does not have delayed effects and resource needs, and (iv) his characterization is in logic (Situation calculus) while ours requires a simpler background of sets and functions. The thesis [15] is perhaps one of the latest report on the work at Linkoping that is related to this paper. Among the main differences between the work there and this paper are: (i) they do not consider continuous effects (an early paper by Sandewall [22] does), and (ii) theirs is a logic based characterization with no notion of states and transition functions. The recent paper [10] gives a language for planning with actions similar to the ones in this paper. The language is divided into 4 levels and continuous effects are in level 4. The semantic characterization in [9] is only up to level 3 with a pointer that a semantics of level 4 is defined based on hybrid automata. Besides this, the main difference between the characterization there and ours is that their characterization is only for planning while ours (as we will elaborate in the full paper) is for planning and other reasoning about action tasks. Their notion of states is much simpler than ours and does not record obligations. Hence, it surely won't lead to a Markovian transition function.

In terms of future work, we plan to have a more thorough comparison with related work. In particular we plan to investigate specific subclasses of domain descriptions in our language for which our transition function based semantics matches with the semantics of the above mentioned works and those addressing continuous changes [24] or numeric fluents [16]. We plan to allow for more general observations, as in [5], and conclusion about missing action occurrences from those observations. We also plan to further extend our planner in [6] based on the characterization in this paper and use domain dependent knowledge to expedite planning in such an environment. We would also like to enhance $\mathcal{ADC}$ by adding to it new features such as those for expressing temporal conditions, conditional effects, or state constraints. This will allow the use of $\mathcal{ADC}$ in representing and reasoning about domains in which the effects of actions depend on the value of some fluents temporally and conditionally.

Acknowledgments

The first two authors would like to acknowledge the support of the NASA grant NCC2-1232. The first author is partially supported by a NSF grant 0070463. The second author is also partially supported by a NSF grant EIA-9810732.

References

[1] F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116(1,2):123–191, 2000.

[2] C. Baral. Reasoning about Actions : Nondeterministic effects, Constraints and Qualification. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 2017–2023. Morgan Kaufmann Publishers, San Francisco, CA, 1995.

[3] C. Baral and M. Gelfond. Representing concurrent actions in extended logic programming. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence, Chambery, France*, pages 866–871. Morgan Kaufmann Publishers, San Francisco, CA, 1993.

[4] C. Baral and M. Gelfond. Reasoning about effects of concurrent actions. *Journal of Logic Programming*, 31(1-3):85–117, May 1997.

[5] C. Baral, M. Gelfond, and A. Provetti. Representing Actions: Laws, Observations and Hypothesis. *Journal of Logic Programming*, 31(1-3):201–243, May 1997.

[6] C. Baral, T.C. Son, and L.C. Tuan. Reasoning about actions in presence of resources: applications to planning and scheduling. In *Proceedings of International conference on information technology (to appear)*, 2001.

[7] P. Doherty, J. Gustafsson, L. Karlsson, and J. Kvarnstrom. Tal: Temporal action logics – language specification and tutorial. *Electronic Transactions on Artificial Intelligence*, 3(15), 1998. http://www.ep.liu.se/ej/etai/1998/015.

[8] P. Doherty and J. Kvarnstom. TALplanner: An Empirical Investigation of a Temporal Logic-based Forward Chaining Planner. In *Proceedings of the 6th Int'l Workshop on the Temporal Representation and Reasoning, Orlando, Fl. (TIME'99)*, 1999.

[9] M. Fox and D. Long. PDDL+: An extension to PDDL for expressing temporal planning domains. Technical report, University of Durham, 2001.

[10] M. Fox and D. Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. Technical report, University of Durham, 2001.

[11] M. Gelfond and V. Lifschitz. Representing actions and change by logic programs. *Journal of Logic Programming*, 17(2,3,4):301–323, 1993.

[12] M. Gelfond and V. Lifschitz. Action languages. *ETAI*, 3(6), 1998.

[13] E. Giunchiglia, G. Kartha, and V. Lifschitz. Representing action: indeterminacy and ramifications. *Artificial Intelligence*, 95:409–443, 1997.

[14] E. Giunchiglia and V. Lifschitz. An action language based on causal explanation: preliminary report. In *Proceedings of AAAI 98*, pages 623–630, 98.

[15] J. Gustafsson. Extending Temporal Action Logic. PhD thesis, Linkoping University, 2001.

[16] J. Lee and V. Lifschitz. Additive fluents. In *AAAI Spring Symposium: "Answer Set Programming: Towards Efficient and Scalable Knowledge Representation and Reasoning"*, pages 116–123. AAAI Press, 2001.

[17] N. McCain and M. Turner. Causal theories of action and change. In *Proceedings of the 14th National Conference on Artificial Intelligence*, pages 460–467. AAAI Press, 1997.

[18] N. McCain and M. Turner. A causal theory of ramifications and qualifications. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1978–1984. Morgan Kaufmann Publishers, San Mateo, CA, 95.

[19] D. Nau, Y. Cao, A. Lotem, and H. Muñoz-Avila. Shop: Simple hierarchical ordered planner. In *Proceedings of the 16th International Conference on Artificial Intelligence*, pages 968–973. AAAI Press, 1999.

[20] R. Reiter. Natural actions, concurrency and continuous time in the situation calculus. In L. Aiello, J. Doyle, and S. Shapiro, editors, *KR 96*, pages 2–13, 1996.

[21] R. Reiter. *KNOWLEDGE IN ACTION: Logical Foundations for Describing and Implementing Dynamical Systems.* MIT Press, 2001.

[22] E. Sandewall. Filter preferential entailment for the logic of action in almost continuous worlds. In N. S. Sridharan, editor, *Proceedings of the 11th International Joint Artificial Intelligence*, pages 894–899, 1989.

[23] T.C. Son and C. Baral. Formalizing sensing actions - a transition function based approach. *Artificial Intelligence*, 125(1-2):19–91, January 2001.

[24] M. Thielscher. Modelling actions with ramifications in nondeterministic, concurrent, and continuous domains – and a case study. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI'2000)*, pages 497–502, 2000.

[25] H. Turner. Representing actions in logic programs and default theories. *Journal of Logic Programming*, 31(1-3):245–298, May 1997.

Knowledge Equivalence in Combined Action Theories

Ronald P. A. Petrick
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada M5S 3G4
rpetrick@cs.utoronto.ca

Hector J. Levesque
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada M5S 3G4
hector@cs.utoronto.ca

Abstract

We investigate the relationship between two accounts of knowledge and action in the situation calculus: the Scherl and Levesque (SL) approach that models knowledge with possible worlds, and the Demolombe and Pozos Parra (DP) approach that models knowledge by a set of "knowledge fluents." We construct *combined action theories*: basic action theories that encode a correspondence between an SL and a DP theory. We prove, subject to certain restrictions, that knowledge of fluent literals are provably the same after a sequence of actions. Moreover, this knowledge equivalence extends to a rich class of formulae. These results allow us to translate certain SL theories into equivalent DP theories that avoid the computational drawbacks of possible world reasoning. They also enable us to prove the correctness of the DP treatment of knowledge and action in terms of a possible world specification.

1 INTRODUCTION

Reasoning about *sensing as a form of action*, for the purpose of planning or high-level agent control, requires the ability to reason effectively about knowledge. Conceptually, reasoning about knowledge and action has been extensively studied and is relatively well understood. For example, Moore (1985) shows how the situation calculus can be adapted to knowledge using the accessibility relation over possible worlds (Hintikka, 1962). Scherl and Levesque (1993) extend Reiter's theory of action (Reiter, 2001a) to handle knowledge, thus providing a solution to the frame problem (McCarthy and Hayes, 1969) for knowledge change. A similar approach is explored by Thielscher (2000), where the fluent calculus is extended to include knowledge update axioms that model an agent's changing knowledge. In (Shapiro et al., 2000), sensing actions are generalized to manage possibly inaccurate beliefs in the situation calculus. Reiter (2001b) considers knowledge-based GOLOG programs with sensing actions. In (Baral and Son, 1997) a high-level action description language is presented that models sensing actions and a distinction between the state of the world and the knowledge of the world.

The approaches mentioned above all share a common treatment of knowledge: reasoning about knowledge is understood as reasoning about the accessibility relation over possible worlds, treated as a fluent that changes due to action. Computationally, this approach is not so promising. The difficulty is that determining if a formula is known then means determining if it is true in all of the currently accessible possible worlds. With n atomic formulae, this means that there are potentially 2^n distinguishable worlds to check truth in. In other words, model checking of formulae about knowledge looks as bad as theorem-proving of formulae without knowledge, and theorem-proving of formulae with knowledge looks even worse. Therefore, even if we were to accept that a planner for ordinary actions based on a formalism like the situation calculus could be made practical, the addition of knowledge and sensing, modelled on possible worlds, raises new concerns.

Consequently, it is not too surprising that many of the attempts to construct planners to effectively manage sensing actions that we are aware of (e.g., (Bacchus and Petrick, 1998; Peot and Smith, 1992; Etzioni et al., 1997; Weld et al., 1998; Pryor and Collins, 1996)) have relied either on variants of STRIPS (Fikes and Nilsson, 1971) or special-purpose algorithmic treatments of knowledge. The trouble with these approaches, however, is in separating any formal semantics from the implementation details of the algorithms that the systems are built on. As a result, it is often quite hard to see how the work relates to a logical specification in a more general theory of knowledge and action.

It is possible, however, to formalize a limited concept of knowledge and sensing in a logical language of action like

the situation calculus without using possible worlds. For example, Funge (1998) restricts knowledge to be about the values of real-valued functional fluents (e.g., distance, temperature, height). What is known is characterized not by an accessibility relation defining possible worlds, but rather by a set of upper and lower bounds that define intervals of possible values for these fluents. More qualitatively, Demolombe and Pozos Parra (2000) characterize knowledge of relational formulae by a set of fluents known true or known false. Instead of formalizing how the set of accessible possible worlds changes as the result of action, they propose to formalize how these "knowledge fluents" change individually. Both of these approaches are very attractive for two important reasons: first, the effect of sensing actions on knowledge is now very similar in form to the effect of ordinary actions on other fluents; second, reasoning about this type of knowledge change is now computationally no worse than reasoning about ordinary fluent change.

But what exactly do we give up in these accounts? What exactly is their relationship to the standard possible world one? In this paper, we propose a partial answer to these questions. We consider *combined action theories*, basic action theories that include axioms from both the Scherl and Levesque (henceforth SL) and the Demolombe and Pozos Parra (henceforth DP) theories of knowledge and action. Our combined action theories will encode a correspondence between an SL theory (using possible worlds) and a DP one (using knowledge fluents). We prove, subject to certain restrictions, that this correspondence maintains the property that fluent literals known are provably the same after a sequence of actions. Moreover, we show that this knowledge equivalence extends to a rich class of formulae. These results are important as they allow us to translate certain SL theories into equivalent DP theories that avoid the computational drawbacks of possible world reasoning. Unfortunately, differences in the expressive nature of the two approaches mean that this correspondence is not one to one. Our results do, however, enable us to prove the correctness of the DP treatment of knowledge and action in terms of the standard possible world specification.

The rest of the paper is organized as follows. In Section 2 we review the situation calculus and the SL and DP theories of knowledge and action. In Section 3 we introduce the notion of a combined action theory. In Section 4, we establish knowledge equivalence of fluent literals for certain classes of combined action theories and extend these results to more general first-order formulae. In Section 5 we give a comprehensive example illustrating our approach. Finally, in Section 6 we discuss some of the issues and possible extensions related to our work.

2 BACKGROUND

2.1 SITUATION CALCULUS

The situation calculus (as presented in (Reiter, 2001a)) is a first-order, many-sorted language (with some second-order features), specifically designed for modelling dynamically changing worlds. All changes to the world are the result of named *actions*. A first-order term called a *situation* is used to represent a possible world history (a sequence of actions). A special constant called S_0 indicates the *initial situation*, that is, the situation in which no actions have yet been performed. There is also a distinguished binary function symbol do such that $do(a, s)$ denotes the successor situation resulting from performing action a in situation s. Actions are denoted by function symbols and may be parameterized, while situations are first-order terms. Relations (predicates) with the property that their truth values can change from situation to situation are referred to as (relational) *fluents*.[1] A fluent is denoted by including a situation argument as its last argument, indicating the value of the fluent at that situation.

Domain theories are specified by defining the following axioms:

- For each action A, an *action precondition axiom* of the form
$$Poss(A(\vec{x}), s) \equiv \Pi_A(\vec{x}, s).^2$$
- For each fluent F, a *successor state axiom* of the form
$$F(\vec{x}, do(a, s)) \equiv \gamma_F^+(\vec{x}, a, s) \vee F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a, s),$$
characterizing the conditions under which fluent F is true at situation $do(a, s)$ as a function of situation s. γ_F^+ (similarly γ_F^-) describes all the ways of making F true (false) in the situation $do(a, s)$ by executing a in situation s.
- A set of first-order sentences describing the initial situation that syntactically only mention the situation term S_0.

Together with a set of unique-names axioms for primitive actions and a set of domain-independent foundational axioms (formally defining legal situations), this collection of axioms forms a *basic action theory*.

[1] *Functional fluents* are also permitted but we will restrict our attention to relational fluents only.

[2] Axioms that contain "free" variables can be thought of as being universally quantified from outside the axiom. Also, for simplicity we will assume that $Poss(A(\vec{x}), s) \equiv \textbf{true}$ for each action A. We will omit any discussion of the *Poss* predicate and assume that actions are always executable.

2.2 A K FLUENT IN THE SITUATION CALCULUS

The situation calculus formalism in (Reiter, 2001a) does not distinguish between what is true in a situation and what is known in a situation. Scherl and Levesque (1993) formalize knowledge in the situation calculus by adapting a standard possible worlds model of knowledge as was done by Moore (1985). A binary relation $K(s', s)$ is introduced, read informally as "s' is accessible from s," and treated like any other fluent (the last argument being the "official" situation argument).

Informally, $K(s', s)$ holds when as far as an agent in situation s knows, it could be in situation s'. The expression[3] $\mathbf{Knows}_{SL}(\phi, s)$ is used to state that ϕ is known in situation s, where ϕ is a situation calculus formula with a special situation term "*now*." The notation $\phi[s]$ is used to indicate the formula that results from replacing *now* in ϕ by s. The expression $\mathbf{Knows}_{SL}(\phi, s)$ is then an abbreviation defined by

$$\mathbf{Knows}_{SL}(\phi, s) \stackrel{\text{def}}{=} (\forall s').K(s', s) \supset \phi[s'].$$

As with other relational fluents, the K fluent possibly changes truth values due to action. The effects that actions have on K are encoded by defining a successor state axiom of the form

$$\begin{aligned} K(s'', do(a, s)) \equiv\ & (\exists s').s'' = do(a, s') \wedge K(s', s) \wedge \\ & \forall((a = \alpha_1) \supset (\phi_1[s] \equiv \phi_1[s'])) \wedge \\ & \ldots \\ & \wedge\forall((a = \alpha_n) \supset (\phi_n[s] \equiv \phi_n[s'])). \end{aligned}$$

Here the α_i are *knowledge-producing* or sensing actions that inform the agent whether or not ϕ_i holds. K is updated to reflect the situations now considered possible, depending on the type of action (knowledge-producing or ordinary).

A particular modal logic is modelled by including axioms that place restrictions on the K accessibility relation. For instance, the S4 modal logic is modelled by including reflexivity and transitivity axioms. Scherl and Levesque also show that provided these properties hold of the K relation in initial situations, then the K relation in every situation resulting from an executable sequence of actions will also satisfy the same set of properties.

Finally, a basic action theory must include axioms that define the possible world alternatives for K in the initial situation S_0. These specifications are necessary to define what is known and what is not known initially. To refer to these initial alternative situations, we include the expression $Init(s)$, to indicate "s is an initial situation." Formally, we define $Init(s)$ as the abbreviation

$$Init(s) \stackrel{\text{def}}{=} \neg(\exists a, s')\ s = do(a, s').$$

2.3 KNOWLEDGE FLUENTS IN THE SITUATION CALCULUS

Demolombe and Pozos Parra (2000) present an alternate approach to modelling knowledge in the situation calculus.[4] A modal operator K is introduced and "combined" syntactically with a non-equality fluent literal P to form a *knowledge fluent* KP.[5] Informally, $KP(s)$ is a fluent meaning "P is known to be true in situation s." These modal fluents are used to explicitly model knowledge without manipulating possible worlds, but restrict the expressive power of the representation to knowledge of literals.

For each ordinary fluent F, a pair of modal fluents, KF and $K\neg F$, are defined. In addition to specifying a standard successor state axiom for F, *successor knowledge state axioms* must be given for both KF and $K\neg F$. These axioms have the same form as regular successor state axioms,

$$\begin{aligned} & KF(\vec{x}, do(a, s)) \equiv \\ & \quad \gamma^{+}_{KF}(\vec{x}, a, s) \vee KF(\vec{x}, s) \wedge \neg\gamma^{-}_{KF}(\vec{x}, a, s), \\ & K\neg F(\vec{x}, do(a, s)) \equiv \\ & \quad \gamma^{+}_{K\neg F}(\vec{x}, a, s) \vee K\neg F(\vec{x}, s) \wedge \neg\gamma^{-}_{K\neg F}(\vec{x}, a, s), \end{aligned}$$

but must ensure that knowledge remains consistent. That is, both $KF(\vec{x}, s)$ and $K\neg F(\vec{x}, s)$ cannot hold in the same situation s.

Since knowledge fluents are ordinary situation calculus fluents, a basic action theory must include axioms defining KF and $K\neg F$ at S_0. These axioms formally define what is initially known (or not known) about an ordinary fluent F.

3 PROPERTIES OF COMBINED ACTION THEORIES

As a first step towards relating the two accounts of knowledge, we begin by defining a *combined action theory*, a basic action theory that includes axioms for the K fluent, successor state axioms for ordinary fluents, a set of successor knowledge state axioms for knowledge fluents, and restrictions on the set of initial situations $Init(s)$. A combined action theory will be used to encode a *translation* between

[3] We are freely changing the notation used by Scherl and Levesque.

[4] In (Demolombe and Pozos Parra, 2000), *belief* is modelled in a KD axiom system. We are instead modelling knowledge and have made notational changes to reflect this difference.

[5] We will use the term *fluent literal* to refer to a fluent $F(\vec{x}, s)$ or its negation $\neg F(\vec{x}, s)$, indicating that either form may be used. Similarly, for $KP(\vec{x}, s)$, where P is a fluent literal (of F), we mean the corresponding knowledge fluent $KF(\vec{x}, s)$ or $K\neg F(\vec{x}, s)$.

the SL and DP axioms by specifying the form of the axioms we consider and the relationship between the SL and DP axioms. In this section we concentrate on the translation of knowledge-producing actions and initial situation axioms, but describe in general how the effects of ordinary physical actions are encoded. We will define 5 properties that any combined action theory must satisfy. In Section 4 we will consider *classes* of combined action theories based on certain restrictions to the successor state axioms. These restrictions will allow us to establish an equivalence between the SL and DP forms of knowledge.

We will assume that we have a finite number of knowledge-producing actions, $\alpha_1, \alpha_2, \ldots, \alpha_m$, and a finite number of physical actions, $\beta_1, \beta_2, \ldots, \beta_n$. We will treat each action as being distinct, and the physical actions as being distinct from the knowledge-producing actions.

3.1 REPRESENTATION OF SENSING ACTIONS

A combined action theory will contain a successor state axiom for K that has the standard SL form. The first property we consider imposes additional restrictions on the form of the sensory effects that can be modelled. These constraints will allow us to translate the effects described in K into appropriate successor knowledge state axioms for which we can establish a knowledge equivalence between the SL and DP forms of representation. Even with these restrictions, we will still be able to model a number of interesting sensory effects.

For instance, consider the axiom for K defined by

$$K(s'', do(a,s)) \equiv (\exists s').s'' = do(a,s') \wedge K(s',s) \wedge \varphi_1(a,s,s') \wedge \varphi_2(a,s,s') \wedge \varphi_m(a,s,s').$$

Say φ_1 defines a knowledge-producing action $sense_1$ as

$$\varphi_1(a,s,s') \stackrel{\text{def}}{=} a = sense_1(x) \supset (F(x,s) \equiv F(x,s')).$$

Here $sense_1$ is a simple action that unconditionally senses the truth value of a fluent F for the specified x. A more complex action is given by:

$$\varphi_2(a,s,s') \stackrel{\text{def}}{=} a = sense_2 \supset (\forall x)(F(x,s) \equiv F(x,s')).$$

In this case the action $sense_2$ has a *universal* sensory effect. The universal quantification of x results in the unconditional sensing of F for each possible value of x. One additional type of sensing action is represented by:

$$\varphi_3(a,s,s') \stackrel{\text{def}}{=} a = sense_3(x) \supset ((G(x,s) \equiv G(x,s')) \wedge G(x,s) \supset (F(x,s) \equiv F(x,s'))).$$

The action $sense_3$ has a compound effect: it unconditionally senses the truth value of G (for the specified x) and also *conditionally* senses F, provided G is true. In general this type of sensing allows additional properties about some set of objects to be sensed, contingent on the truth of some initial property. Our representation allows finite "chains" of this type of sensing to be modelled, and also allows situation independent formulae to be specified as conditions.

Formally, we have the following definition of the K axiom.

Property 1 Let $\alpha_1, \alpha_2, \ldots, \alpha_m$ be distinct knowledge-producing action terms. The successor state axiom for the K fluent has the form

$$K(s'', do(a,s)) \equiv (\exists s').s'' = do(a,s') \wedge K(s',s) \wedge \varphi_1(a,s,s') \wedge \varphi_2(a,s,s') \wedge \ldots \wedge \varphi_m(a,s,s').$$

For each φ_i let $F_1, F_2, \ldots, F_l$ be distinct fluents so that

$$\varphi_i \stackrel{\text{def}}{=} \forall(a = \alpha_i(\vec{y}) \supset \psi_1 \wedge \psi_2 \wedge \ldots \wedge \psi_l),$$
$$\psi_j \stackrel{\text{def}}{=} (\forall \vec{z}).C_j(\vec{y},\vec{z},s) \supset F_j(\vec{y},\vec{z},s) \equiv F_j(\vec{y},\vec{z},s').$$

ψ_j describes an effect with *condition* C_j and fluent F_j. C_1 is a situation independent formula. For $j > 1$, C_j is either a situation independent formula or a conjunction of the form:

$$C_{j-1}(\vec{y},\vec{z},s) \wedge [\neg]F_{j-1}(\vec{y},\vec{z},s),$$

where C_{j-1} is the condition associated with ψ_{j-1}.

3.2 SUCCESSOR KNOWLEDGE STATE AXIOMS

For every ordinary fluent F in our SL theory, our combined action theory will include a pair of DP successor knowledge state axioms for $KF, K\neg F$. The second property we consider concerns the form of these axioms which encode all the effects of actions on the agent's knowledge of the fluent F. In other words, this encoding specifies the translation of the SL successor state axioms for K and F into DP axioms. Since we require a translation that preserves an equivalence with the effects described by the SL theory, we must consider two different types of effects: the effects of physical actions and the effects of knowledge-producing actions.

For physical actions, the equivalence is achieved by converting ordinary successor state actions into "knowledge fluent versions," through syntactic changes to fluent literals. All references to P in $\gamma_F^{\pm}$ are changed to KP without changing the underlying structure of $\gamma_F^{\pm}$ (i.e., the logical connectives). In the case of a situation independent formula, the conversion leaves the formula unchanged. (In Section 4 we will apply this syntactic conversion to restricted successor state axioms.)

For knowledge-producing actions, the equivalence depends on extracting the separate effects of all knowledge-producing actions on a particular fluent (defined in K) and

packaging them together into the pair of corresponding successor knowledge state axioms. The appropriate components of the K axiom (i.e., the specific effects that sense the fluent F) are incorporated into the successor knowledge state axioms, maintaining the same structure of the action terms and conditions on conditional effects that are defined for K. Any explicit quantification becomes implicitly quantified in the successor knowledge state axiom.

Consider the actions $sense_1$, $sense_2$, and $sense_3$, defined in Section 3.1. Assuming no other actions sense F we can generate the corresponding pair of successor knowledge state axioms for $KF, K\neg F$:

$$\begin{aligned} &KF(x, do(a,s)) \equiv \\ &\quad (\gamma_F^+)^K \vee ((a = sense_1(x) \vee a = sense_2 \vee \\ &\quad (a = sense_3(x) \wedge G(x,s))) \wedge F(x,s)) \vee \\ &\quad KF(x,s) \wedge \neg(\gamma_F^-)^K, \\ &K\neg F(x, do(a,s)) \equiv \\ &\quad (\gamma_F^-)^K \vee ((a = sense_1(x) \vee a = sense_2 \vee \\ &\quad (a = sense_3(x) \wedge G(x,s))) \wedge \neg F(x,s)) \vee \\ &\quad K\neg F(x,s) \wedge \neg(\gamma_F^+)^K. \end{aligned}$$

($\gamma_F^{\pm}$ is defined in the successor state axiom for F.) Note that the explicit universal quantification in φ_2 is now expressed implicitly in the successor knowledge state axioms.

We formally define the translation of successor state axioms to successor knowledge state axioms as follows.

Property 2 For each ordinary fluent F, the successor knowledge state axioms for knowledge fluents $KF, K\neg F$ are of the form

$$\begin{aligned} &KF(\vec{x}, do(a,s)) \equiv \\ &\quad \gamma_{KF}^+(\vec{x}, a, s) \vee KF(\vec{x}, s) \wedge \neg\gamma_{KF}^-(\vec{x}, a, s), \\ &K\neg F(\vec{x}, do(a,s)) \equiv \\ &\quad \gamma_{KF}^-(\vec{x}, a, s) \vee K\neg F(\vec{x}, s) \wedge \neg\gamma_{KF}^+(\vec{x}, a, s), \end{aligned}$$

and $\gamma_{KF}^{\pm}(\vec{x}, a, s)$ has the form $(\gamma_F^{\pm})^K \vee \xi_F^{\pm}$.

$(\gamma_F^{\pm})^K$ is structurally identical to $\gamma_F^{\pm}$ with the exception that every fluent literal P is syntactically replaced by KP. $\xi_F^{\pm}$ has the form

$$\bigvee_{\langle i,j\rangle \in S_F} (a = \alpha_i(\vec{y}) \wedge C_j(\vec{y}, \vec{z}, s) \wedge \pm F(\vec{y}, \vec{z}, s)),$$

where S_F is defined for each fluent F as

$$S_F \stackrel{\text{def}}{=} \{\langle i,j\rangle \mid \varphi_i \text{ has an effect } \psi_j \text{ with condition } C_j \text{ and fluent } F \text{(defined in the } K \text{axiom)}\}.$$

S_F indicates the components of the K axiom that sense F. $\alpha(\vec{y})$ and C_j are structurally identical to those defined in Property 1. If no knowledge-producing action has an effect on a fluent F, then $\gamma_{KF}^{\pm}$ reduces to $(\gamma_F^{\pm})^K$.

3.3 CONSTRAINTS ON INITIAL SITUATIONS

We now consider the three final properties required of a combined action theory, dealing with initial situations. First, since we are modelling knowledge we require that a reflexivity restriction hold of the K fluent. As shown in (Scherl and Levesque, 1993) we only require that this property hold of initial situations for it to hold for all situations. A consequence of reflexivity, however, is that our initial knowledge must correspond correctly to the initial values of ordinary fluents (i.e., the way the real world is initially configured). Formally, we require the following conditions hold:

Property 3 *(reflexivity of K)*

$$\begin{aligned} &\Sigma \models (\forall s).Init(s) \supset K(s,s), \\ &\Sigma \models (\forall s)(\forall \vec{x}).Init(s) \supset \\ &\qquad \mathbf{Knows}_{SL}(P(\vec{x}, now), s) \supset P(\vec{x}, s), \end{aligned}$$

for every fluent literal P.

Second, we require a knowledge equivalence for initial situations to ensure that we begin with literal-based knowledge that is identical in terms of both the SL (using K and possible worlds) and DP (using knowledge fluents) forms of representation. Our goal in Section 4 will be to show that this equivalence is preserved through action, subject to certain restrictions that we place on the form of the combined action theory. Formally, we require the following property:

Property 4 *(initial knowledge equivalence)* For every fluent literal P,

$$\begin{aligned} &\Sigma \models (\forall s)(\forall \vec{x}).Init(s) \supset \\ &\qquad \mathbf{Knows}_{SL}(P(\vec{x}, now), s) \equiv KP(\vec{x}, s). \end{aligned}$$

Finally, we require a strong restriction on our initial knowledge to ensure that we can "break apart" any knowledge of disjunctions to reason about the knowledge of the individual disjuncts. In general, SL theories can model knowledge of disjunctions without requiring knowledge of individual disjuncts.

Example 1 Consider the following axioms:

$$\begin{aligned} &F(do(a,s)) \equiv F(s), \\ &G(do(a,s)) \equiv G(s), \\ &K(s'',s) \equiv (\exists s').s'' = do(a,s') \wedge K(s',s) \wedge \\ &\quad (a = sense \supset ((F(s) \vee G(s)) \equiv (F(s') \vee G(s')))), \\ &(\exists s_1, s_2, s_3, s_4).K(s_1, S_0) \wedge K(s_2, S_0) \wedge K(s_3, S_0) \wedge \\ &\quad K(s_4, S_0) \wedge F(s_1) \wedge G(s_1) \wedge F(s_2) \wedge \neg G(s_2) \wedge \\ &\quad \neg F(s_3) \wedge G(s_3) \wedge \neg F(s_4) \wedge \neg G(s_4). \end{aligned}$$

Initially, nothing is known about the fluents F and G, however, in the situation $S = do(sense, S_0)$ we have that $\mathbf{Knows}_{SL}((F(now) \vee G(now)), S)$ holds, but neither $\mathbf{Knows}_{SL}(F(now), S)$ nor $\mathbf{Knows}_{SL}(G(now), S)$ hold.

In DP theories, however, the representation is restricted to knowledge of fluent literals. Thus, we require a disjunctive knowledge restriction to ensure that we can establish an equivalence of literal-based knowledge that can be preserved after a sequence of actions.

Property 5 *(initial disjunctive knowledge)* For all fluent literals $P_1, P_2, \ldots, P_k$ that are not complementary,[6] and any ground sequence of actions $\vec{A}_1, \vec{A}_2, \ldots, \vec{A}_k$,

$$\Sigma \models (\forall s)(\forall \vec{x}).Init(s) \supset \mathbf{Knows}_{SL}(\bigvee_{i=1}^{k} P_i(\vec{x}, do(\vec{A}_i, now)), s) \equiv \bigvee_{i=1}^{k} \mathbf{Knows}_{SL}(P_i(\vec{x}, do(\vec{A}_i, now)), s).$$

This property not only specifies that we can break apart "immediate" disjunctive knowledge (e.g., formulae such as $\mathbf{Knows}_{SL}(P(now), S_0)$ that include *now* but no other action terms) but that we can also do the same for knowledge of "future" disjunctions (e.g., formulae such as $\mathbf{Knows}_{SL}(P(do(\vec{A}, now)), S_0)$ that include an action sequence $\vec{A}$). It is this second condition that is important for ensuring a literal-based knowledge equivalence can be maintained through action. It also means, however, that we impose strong restrictions on the structure of our initial situations. This issue will be discussed further in Section 4.4.

The strength of Property 5 allows us to extend it to hold for all situations, not just initial situations, given a successor state axiom for K of the form in Property 1.

Theorem 1 *Let Σ be a basic action theory that satisfies Properties 1 and 5. Then Property 5 holds for all situations, not just $Init(s)$.*

Proof (By induction over situations) The base case follows directly from Property 5. In the induction step we consider the two types of actions. For physical actions, using the definitions of $\mathbf{Knows}_{SL}$, the K axiom from Property 1, and the fact that if $\mathbf{Knows}_{SL}(P(\vec{c}, do(A, now)), s)$ holds then $\mathbf{Knows}_{SL}(P(\vec{c}, now), do(A, s))$ holds for all s and A, the result quickly follows. For sensing actions, the K axiom specifies that all knowledge-producing effects reduce to sensing the truth of individual fluent literals, thus preserving the required property. ■

This property will also be required to extend our equivalence results to more general formulae (see Section 4.3).

3.4 COMBINED ACTION THEORIES

We are now able to give a formal definition of a combined action theory, based on the properties described in Sections 3.1–3.3.

[6] That is, we cannot include both P_i and $\neg P_i$.

Definition 1 A *combined action theory* Σ is a basic action theory that satisfies Properties 1–5.

Note that our definition does not specifically define the form of the ordinary successor state axioms (with the exception of K). It does, however, specify how such axioms will be converted to successor knowledge state axioms. In the next section we focus on the restrictions we require of successor state axioms.

4 KNOWLEDGE EQUIVALENCE IN COMBINED ACTION THEORIES

In general, our combined action theory alone is not enough to establish a knowledge equivalence between SL and DP theories, even with the strong restrictions placed on the initial situations.

Example 2 Consider the following axioms:

$$F(do(a, s)) \equiv (a = A \wedge \neg F(s)) \vee F(s),$$
$$\neg\mathbf{Knows}_{SL}(F(now), S_0),$$
$$\neg\mathbf{Knows}_{SL}(\neg F(now), S_0).$$

In terms of literal-based knowledge, nothing is known about F at S_0. However, in the situation $S = do(A, S_0)$, $\mathbf{Knows}_{SL}(F(now), S)$ holds. In this case, knowledge of F at S does not depend on knowing individual literals (i.e., knowing F holds at all possible worlds). Rather, it involves a property that is true of each possible world, in this case a "hidden" tautology (i.e., $F \vee \neg F$ holds at all possible worlds).[7] It is this general representation of knowledge, allowable in SL theories, that poses a problem for DP theories since DP theories are restricted to knowledge of literals and unable to encode such knowledge.

Thus, to ensure a translation between the SL and DP forms of knowledge can be achieved, we are faced with the task of either first removing the hidden logical constraints (such as tautologies) from a theory and constructing a new, logically equivalent theory, or restricting the form of the theories we consider to avoid such issues entirely. We adopt the latter approach and consider restrictions to the form of the successor state axioms that allow us to define *classes* of combined action theories.

4.1 CONTEXT FREE THEORIES

The first class of combined action theories we investigate is formed by restricting our successor state axioms to be *context free*:

[7] Note that our disjunctive knowledge restriction does not forbid this.

Definition 2 (following (Lin and Reiter, 1997)) A successor state axiom for a fluent F is *context free* iff it has the form

$$F(\vec{x}, do(a, s)) \equiv \gamma_F^+(\vec{x}, a) \vee F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a),$$

where $\gamma_F^+(\vec{x}, a)$ and $\gamma_F^-(\vec{x}, a)$ are situation independent formulae whose free variables are among those in $\vec{x}$, a.

Definition 3 A *context free combined action theory* Σ is a combined action theory with the property that successor state axioms for ordinary fluents F are context free (i.e., γ_F^+ and γ_F^- are situation independent).

A context free successor state axiom for a fluent F prohibits any references to fluents in $\gamma_F^{\pm}$. Even with these restrictions, axioms of this form are common. Quantification is still permitted, provided the scope of the quantifiers only range over the situation independent formulae. By requiring successor state axioms be context free, however, we are also placing restrictions on the form of the successor knowledge state axioms (at least the part determined by the effects of physical actions on a fluent). In this case, the physical effect portion of $\gamma_{KF}^{\pm}$ (i.e., $(\gamma_F^{\pm})^K$) will also be context free and, in fact, identical to $\gamma_F^{\pm}$.

For instance, consider the following context free successor state axiom for an ordinary fluent *broken*:

$$broken(x, do(a, s)) \equiv (a = drop(x) \wedge fragile(x)) \vee broken(x, s) \wedge \neg(a = repair(x)).$$

Assuming that no actions also sense *broken*, Definition 3 allows us to generate a corresponding pair of successor knowledge state axioms based solely on the definition of $\gamma_{broken}^{\pm}$ given above. The resulting axioms for *Kbroken*, *K¬broken* (following the form specified in Definition 1) are also context free:

$$Kbroken(x, do(a, s)) \equiv (a = drop(x) \wedge fragile(x)) \vee Kbroken(x, s) \wedge \neg(a = repair(x)),$$
$$K\neg broken(x, do(a, s)) \equiv a = repair(x) \vee K\neg broken(x) \wedge \neg(a = drop(x) \wedge fragile(x)).$$

These restrictions enable us to establish our first equivalence result between the SL and DP definitions of knowledge, not just for initial situations, but for every situation:

Theorem 2 *Let Σ be a context free combined action theory. Then for any fluent literal P,*

$$\Sigma \models (\forall s)(\forall \vec{x}).\mathbf{Knows}_{SL}(P(\vec{x}, now), s) \equiv KP(\vec{x}, s).$$

Proof (By induction over situations) The base case follows directly from Definition 1. In the induction step we consider sensing and physical actions separately. For sensing actions, the basic argument follows from the form of the successor state and successor knowledge state axioms: successor state axioms leave the truth of all ordinary fluents unchanged and the translation in Definition 1 ensures that the corresponding components of the successor state and successor knowledge state axiom will necessarily hold in both axioms if they hold in one axiom. Reflexivity and the definition of $\mathbf{Knows}_{SL}$ ensures we can establish the truth of fluents in the "real" situation. For physical actions, we use the property that $\Sigma \models (\forall s).\mathbf{Knows}_{SL}(\psi, s) \equiv \psi$, when ψ is situation independent. Since $(\gamma_F^{\pm})^K$ and $\gamma_F^{\pm}$ are identical situation independent formulae, and the induction assumption lets us convert between SL and DP knowledge for fluent literals, the result quickly follows from the correspondence between the successor state and successor knowledge state axioms. ■

This result means that as far as knowledge of fluent literals is concerned, the SL and DP accounts are identical and will remain identical after any executable sequence of actions. In practical terms this means that we can exchange an SL theory based on possible worlds for a corresponding DP theory based on knowledge fluents (e.g., the DP axioms for *Kbroken*, *K¬broken* replace the SL axioms for *broken* and K), provided we can accept the limitation of literal-based knowledge.

4.2 LITERAL-BASED THEORIES

We now consider a much more expressive class of combined action theories, formed by extending our successor state axioms to include fluent literals.

Our definition of a literal-based combined action theory forces $\gamma_F^{\pm}$ to be described in a disjunctive normal form, subject to certain restrictions. As with the context free case, quantifiers are allowed, provided their scope only ranges over the situation independent formulae. (An exception is made for variables $\vec{y}$ that appear as parameters to the action.) Additional restrictions ensure that no problematic logical constraints (such as tautologies) arise.

Definition 4 A *literal-based combined action theory* Σ is a combined action theory where the successor state axioms for ordinary fluents F have the property that

$$\gamma_F^{\pm}(\vec{x}, a, s) \stackrel{\text{def}}{=} \bigvee_{i=1}^{k} \pi_i(\vec{x}, a, s).$$

For each π_i, let $P_1, P_2, \ldots, P_l$ be fluent literals, ψ_i a situation independent formula, β_i a physical action term, and $\vec{y}_i$ a vector of variables (possibly empty) so that

$$\pi_i(\vec{x}, a, s) \stackrel{\text{def}}{=} (\exists \vec{y}_i).a = \beta_i(\vec{x}, \vec{y}_i) \wedge \psi_i(\vec{x}, \vec{y}_i, a) \wedge P_1(\vec{x}, \vec{y}_i, s) \wedge P_2(\vec{x}, \vec{y}_i, s) \wedge \ldots \wedge P_l(\vec{x}, \vec{y}_i, s),$$

where $\vec{y}_i$ must be a parameter of β_i, and $\neg F$ (similarly F) can't be mentioned in γ_F^+ (similarly γ_F^-). We also require

the following property hold of every $\pi_i, \pi_j, i \neq j$: for every substitution σ of $\vec{x}, a, \vec{y}_i, \vec{y}_j$ so that

$$\Sigma \models (\beta_i(\vec{x}, \vec{y}_i) = \beta_j(\vec{x}, \vec{y}_j) \wedge \psi_i(\vec{x}, \vec{y}_i) \wedge \psi_j(\vec{x}, \vec{y}_j))\sigma,$$

then for all fluent literals P in π_i and R in π_j, (i) if π_i is in γ_F^+ and π_j is in γ_F^-: $P\sigma$ cannot unify with $R\sigma$, otherwise (ii) $P\sigma$ cannot unify with $\neg R\sigma$.

In practice, these constraints are conservative, yet many of the successor state axioms that occur in the literature can be converted to this form. For instance, consider the following successor state axiom for a fluent *holding*:

$$\begin{aligned} \mathit{holding}(x, do(a, s)) \equiv\ & (a = \mathit{pickup}(x) \vee \\ & (\exists y).a = \mathit{pickup}(y) \wedge \mathit{in}(x, y, s)) \vee \\ & \mathit{holding}(x, s) \wedge \neg(a = \mathit{dropall}). \end{aligned}$$

Following our definition, successor knowledge state axioms must encode knowledge fluent versions of the ordinary successor state axioms. References to fluent literals P in $\gamma_F^{\pm}$ are syntactically replaced by references to KP in $(\gamma_F^{\pm})^K$. For the *holding* example, assuming no actions also sense *holding*, we have the following successor knowledge state axioms:

$$\begin{aligned} \mathit{Kholding}(x, do(a, s)) \equiv\ & (a = \mathit{pickup}(x) \vee \\ & (\exists y).a = \mathit{pickup}(y) \wedge \mathit{Kin}(x, y, s)) \vee \\ & \mathit{Kholding}(x, s) \wedge \neg(a = \mathit{dropall}), \\ K\neg\mathit{holding}(x, do(a, s)) \equiv\ & a = \mathit{dropall} \vee \\ & K\neg\mathit{holding}(x, s) \wedge \neg(a = \mathit{pickup}(x) \vee \\ & (\exists y).a = \mathit{pickup}(y) \wedge \mathit{Kin}(x, y, s)). \end{aligned}$$

This translation allows our equivalence result for context free theories to be extended to literal-based theories as well.

Theorem 3 *Let Σ be a literal-based combined action theory. Then for any fluent literal P,*

$$\Sigma \models (\forall s)(\forall \vec{x}).\mathbf{Knows}_{SL}(P(\vec{x}, now), s) \equiv KP(\vec{x}, s).$$

Proof (By induction over situations) The base case follows directly from Definition 1. In the induction step, consider two types of actions. For sensing actions, the proof is the same as in Theorem 2. For physical actions, in the if direction we repeatedly choose disjunctions of fluent literals from the successor state axiom that must be known. Our restrictions in Definition 4 allow us to apply Theorem 1 to break apart this knowledge into component parts. This process terminates with the fluents in some component of the axiom being known individually. Using the form of the corresponding successor knowledge state axioms and the induction assumption we are able to establish the result. In the only-if direction, the induction assumption applied to the appropriate components of the successor knowledge state axioms relates the knowledge fluents to SL knowledge. The result then follows by considering the form of the successor state axiom and its corresponding translation described in Definition 1. ■

4.3 EXTENDING KNOWLEDGE EQUIVALENCE TO FIRST-ORDER FORMULAE

Up to this point we have only established a knowledge equivalence between SL and DP theories for fluent literals. We now seek to extend that equivalence to account for more general first-order formulae. We begin by defining the expression $\mathbf{Knows}_{DP}(\phi, s)$, to indicate that ϕ is known (in the DP sense) in situation s:

Definition 5 Let F be a fluent and let ϕ and ψ be first-order formulae that don't mention K or any knowledge fluents $KF, K\neg F$. Then

1. $\mathbf{Knows}_{DP}(\phi, s) \stackrel{\text{def}}{=} \phi$, if ϕ is situation independent,
2. $\mathbf{Knows}_{DP}(F(\vec{x}), s) \stackrel{\text{def}}{=} KF(\vec{x}, s)$,
3. $\mathbf{Knows}_{DP}(\neg F(\vec{x}), s) \stackrel{\text{def}}{=} K\neg F(\vec{x}, s)$,
4. $\mathbf{Knows}_{DP}(\neg\neg\phi, s) \stackrel{\text{def}}{=} \mathbf{Knows}_{DP}(\phi, s)$,
5. $\mathbf{Knows}_{DP}(\phi \wedge \psi, s) \stackrel{\text{def}}{=}$
   $\mathbf{Knows}_{DP}(\phi, s) \wedge \mathbf{Knows}_{DP}(\psi, s)$,
6. $\mathbf{Knows}_{DP}(\neg(\phi \wedge \psi), s) \stackrel{\text{def}}{=}$
   $\mathbf{Knows}_{DP}(\neg\phi, s) \vee \mathbf{Knows}_{DP}(\neg\psi, s)$,
7. $\mathbf{Knows}_{DP}((\forall \vec{x}).\psi, s) \stackrel{\text{def}}{=} (\forall \vec{x}).\mathbf{Knows}_{DP}(\phi, s)$,
8. $\mathbf{Knows}_{DP}(\neg(\forall \vec{x}).\psi, s) \stackrel{\text{def}}{=} (\exists \vec{x}).\mathbf{Knows}_{DP}(\neg\phi, s)$.

Using Definition 5 we can now refer to DP knowledge beyond that of simple knowledge fluents. Since our definition of $\mathbf{Knows}_{SL}$ can already be applied to such general formulae, a reasonable question to ask is whether our equivalence results can also be extended to a more general class of formulae. We offer a partial answer to this question. First, we extend our results to disjunctive formulae:

Lemma 1 *Let Σ be a context free or literal-based combined action theory. Let ϕ be a disjunction of non-complementary ground fluent literals. Then*

$$\Sigma \models (\forall s).\mathbf{Knows}_{SL}(\phi, s) \equiv \mathbf{Knows}_{DP}(\phi, s).$$

Proof Let ϕ be a disjunction of the ground fluent literals $P_1, P_2, \ldots, P_k$ and let s be any situation. By Theorem 1:
$\Sigma \models \mathbf{Knows}_{SL}(\phi, s) \equiv \bigvee_{i=1}^{k} \mathbf{Knows}_{SL}(P_i(\vec{c}_i, now), s)$.
Since Σ is a context free (similarly, literal-based) combined action theory, by Theorem 2 (similarly, Theorem 3):
$\Sigma \models \bigvee_{i=1}^{k} \mathbf{Knows}_{SL}(P_i(\vec{c}_i, now), s) \equiv$
$\bigvee_{i=1}^{k} \mathbf{Knows}_{DP}(P_i(\vec{c}_i, now), s)$.
Now, by applying Definition 5 we obtain the desired result:
$\Sigma \models \bigvee_{i=1}^{k} \mathbf{Knows}_{DP}(P_i(\vec{c}_i, now), s) \equiv \mathbf{Knows}_{DP}(\phi, s)$.
■

Although Lemma 1 requires that a disjunctive formula be free of tautologies (in order to make use of our disjunctive knowledge restriction), we can use this lemma to establish the following general equivalence:

Theorem 4 *Let Σ be a context free or literal-based combined action theory. Let ϕ be any ground, quantifier-free first-order formula without K or any knowledge fluents. Then, there is a logically equivalent formula ϕ' such that*

$$\Sigma \models (\forall s).\mathbf{Knows}_{SL}(\phi', s) \equiv \mathbf{Knows}_{DP}(\phi', s).$$

Proof The ϕ' in question will be the conjunction of the non-tautologous prime implicates of ϕ. Thus, $\models \phi \equiv \phi'$. Denote the prime implicates by $\pi_1, \pi_2, \ldots, \pi_k$ and let s be any situation. Since:
$\Sigma \models \mathbf{Knows}_{SL}(\bigwedge_{i=1}^{k} \pi_i, s) \equiv \bigwedge_{i=1}^{k} \mathbf{Knows}_{SL}(\pi_i, s)$
(a property of $\mathbf{Knows}_{SL}$), the clausal form of the prime implicates allows us to apply Lemma 1 so that we have:
$\Sigma \models \bigwedge_{i=1}^{k} \mathbf{Knows}_{SL}(\pi_i, s) \equiv \bigwedge_{i=1}^{k} \mathbf{Knows}_{DP}(\pi_i, s)$.
Now, applying Definition 5 establishes the result:
$\Sigma \models \bigwedge_{i=1}^{k} \mathbf{Knows}_{DP}(\pi_i, s) \equiv \mathbf{Knows}_{DP}(\bigwedge_{i=1}^{k} \pi_i, s)$.
■

This theorem illustrates that our equivalence results can be extended to the class of ground, quantifier-free formulae. In particular, a sentence can be formulated in such a way that it is known in the SL sense (with possible worlds) iff it is known in the DP sense (with knowledge fluents). Furthermore, we believe that this equivalence can be extended to include formulae containing quantifiers. For instance, the techniques used in (Levesque, 1998) could be adopted to deal with such formulae, by restricting them to be in a normal form.

While Theorem 4 does not allow quantification in general, provided we ensure that the scope of quantifiers range only over situation independent formulae (i.e., "closed" situation independent formulae) we can consider a simple extension to our equivalence results:

Corollary 1 *Let Σ be a context free or literal-based combined action theory. Let ϕ be any first-order sentence, without K or any knowledge fluents, whose quantifiers only range over situation independent formulae. Then, there is a logically equivalent formula ϕ' such that*

$$\Sigma \models (\forall s).\mathbf{Knows}_{SL}(\phi', s) \equiv \mathbf{Knows}_{DP}(\phi', s).$$

Proof Put ϕ into a conjunctive normal form, keeping any situation independent formula closed. Break apart the conjunctions into knowledge of the component parts. Since $\Sigma \models (\forall s).\mathbf{Knows}_{SL}(\psi, s) \equiv \psi$, when ψ is a situation independent formula (a property of $\mathbf{Knows}_{SL}$), the result quickly follows from Theorem 4 and Definition 5. ■

4.4 DISJUNCTIVE KNOWLEDGE AND INITIAL SITUATIONS

Our definition of a combined action theory enforces a strong property on disjunctive knowledge, namely that disjunctions (both immediate and future) can be broken apart into knowledge of the individual disjuncts. Moreover, provided that this property holds in all initial situations, it will also hold in all subsequent situations, independent of the combined action theory. But what exactly does this property tell us about the structure of S_0 and other initial situations? Is such a strong property necessary?

We begin by considering a much less restrictive property about disjunctive knowledge:

Definition 6 Let Σ be a basic action theory. A situation s is said to satisfy the *weak disjunctive knowledge property* if for all fluent literals $P_1, P_2, \ldots, P_k$ that are not complementary,

$$\Sigma \models (\forall \vec{x}).\mathbf{Knows}_{SL}(\bigvee_{i=1}^{k} P_i(\vec{x}, now), s) \equiv \bigvee_{i=1}^{k} \mathbf{Knows}_{SL}(P_i(\vec{x}, now), s).$$

With this weaker form of disjunctive knowledge we no longer require constraints on "future" disjunctions, just immediate ones. It turns out that for the class of context free theories such a property is sufficient to maintain our equivalence results.

Theorem 5 *Let Σ be defined as in Definition 3 with the (strong) disjunctive knowledge property replaced with the weak disjunctive knowledge property on initial situations. Then, (i) the weak disjunctive knowledge property holds for all situations, and (ii) the equivalence results of Sections 4.1 and 4.3 extend to Σ.*

Proof The proof of (i) is straight-forward by induction over situations, using the form of the translation in Definition 1 when we have context free successor state axioms, and the property that $\Sigma \models (\forall s).\mathbf{Knows}_{SL}(\psi, s) \equiv \psi$, when ψ is situation independent. For (ii), the proofs carry over from Sections 4.1 and 4.3 with all references to Theorem 1 replaced with references to Theorem 5(i). ■

Thus, for context free theories at least we need only be concerned about immediate disjunctions in the initial situation (i.e., those that only mention *now* and no other action terms). This weaker notion of disjunctive knowledge, however, is not necessarily preserved if we consider non-context free theories, even literal-based ones.

Example 3 Consider the following axioms:

$$F(do(a, s)) \equiv (a = A \wedge G(s)) \vee F(s),$$
$$G(do(a, s)) \equiv G(s),$$
$$(\exists s_1, s_2, s_3, s_4).K(s_1, S_0) \wedge K(s_2, S_0) \wedge K(s_3, S_0) \wedge K(s_4, S_0) \wedge F(s_1) \wedge G(s_1) \wedge F(s_2) \wedge \neg G(s_2) \wedge \neg F(s_3) \wedge G(s_3) \wedge \neg F(s_4) \wedge \neg G(s_4).$$

Nothing is known initially about F and G. (The specification of initial situations means that the weak disjunctive

knowledge property holds of S_0.) In the situation $S = do(A, S_0)$, however, we have that $\mathbf{Knows}_{SL}((F(now) \vee \neg G(now)), S)$ holds, but neither $\mathbf{Knows}_{SL}(F(now), S)$ nor $\mathbf{Knows}_{SL}(\neg G(now), S)$ hold.

Thus, by considering even slightly more complex successor state axioms the weaker notion of disjunctive knowledge can quickly fail. Since we require such a property hold in order to establish our equivalence results, this motivates the need for our stronger restriction.

What this property does *not* provide, however, is an efficient method of detecting all the necessary conditions that must hold of an initial situation. In constructing an SL theory one must potentially consider disjunctions that arise from *any* sequence of actions and make sure that the appropriate knowledge is encoded in the initial situations. For instance, in Example 3 we would require that $\mathbf{Knows}_{SL}(F(do(A, now)), S_0)$ or $\mathbf{Knows}_{SL}(\neg G(do(A, now)), S_0)$ hold of S_0.

4.5 NON-EQUIVALENCE OF SL AND DP THEORIES

While we have been able to correlate the SL and DP approaches for an expressive class of theories, the equivalence of SL and DP theories is not one-to-one. Clearly, there exist SL theories without equivalent DP formulations (e.g., Example 1). The converse is also true. Depending on the form of the successor knowledge state axioms, DP theories can be modelled so that knowledge fluents evolve independent of ordinary fluents. Consequently, we can construct DP theories that manipulate knowledge in a way that cannot be easily reproduced in a standard SL theory.

Example 4 Consider the following axioms:

$$F(do(a, s)) \equiv F(s),$$
$$KF(do(a, s)) \equiv KF(do(a, s)) \wedge \neg(a = forget),$$
$$K\neg F(do(a, s)) \equiv K\neg F(do(a, s)) \wedge \neg(a = forget),$$
$$KF(F, S_0).$$

In the situation $S = do(forget, S_0)$, both $\neg KF(S)$ and $\neg K\neg F(S)$ hold. Thus, *forget* produces a *knowledge reducing* effect without changing any ordinary fluents. Such an action cannot be modelled directly in a standard SL theory (see the theorems concerning *memory* in (Scherl and Levesque, 1993)).

To make our equivalence more encompassing, one possibility is to extend the SL theory. For instance, a richer representation that allows actions such as *forget* to be modelled at the possible world level could provide a closer correspondence to the DP theory. We are currently investigating such an approach as well as alternate theories that could subsume the SL approach altogether (see Section 6).

5 AN EXAMPLE

One of our main objectives has been to provide a means of translating certain SL theories into equivalent DP theories that avoid the use of possible worlds. We now illustrate our approach with an example from the UNIX domain that involves both ordinary and knowledge-producing actions.

Consider the following UNIX-style domain involving two fluents, *indir* and *readable*, and two actions, *ls* and *mv*. The fluent $indir(f, d, s)$ can be understood as "file f is in directory d in situation s." The fluent $readable(f, s)$ indicates that "file f is readable in situation s." The action $mv(f, d', d)$ is an ordinary (physical) action that has the effect of moving file f from directory d' to directory d. The action $ls(d)$ is a knowledge-producing action that provides information about the files in directory d. We encode the successor state axioms in our SL theory as follows:

$$indir(f, d, do(a, s)) \equiv$$
$$((\exists d').a = mv(f, d', d) \wedge d \neq d' \wedge indir(f, d', s)) \vee$$
$$indir(f, d, s) \wedge \neg((\exists d').a = mv(f, d, d') \wedge d \neq d'),$$

$$readable(f, do(a, s)) \equiv readable(f, s),$$

$$K(s'', do(a, s)) \equiv (\exists s').s'' = do(a, s') \wedge K(s', s) \wedge$$
$$(\exists d).(a = ls(d) \supset$$
$$(\forall f)\ (indir(f, d, s) \equiv indir(f, d, s')) \wedge$$
$$(\forall f)\ (indir(f, d, s) \supset$$
$$(readable(f, s) \equiv readable(f, s')))).$$

Our successor state axiom for K encodes two types of knowledge-producing effects for *ls*. First, it encodes a universal effect: *ls* senses the files f that are in directory d (i.e., all f that satisfy $indir(f, d, s)$). It also encodes a type of conditional sensing effect: besides sensing the contents of the directory, *ls* also senses the readability *of the files that are in directory* d (i.e., $readable(f, s)$ for all f such that $indir(f, d, s)$ is true). Using Definitions 1 and 4, we can translate the SL axioms into corresponding DP axioms:

$$Kindir(f, d, do(a, s)) \equiv$$
$$((\exists d').a = mv(f, d', d) \wedge d \neq d' \wedge Kindir(f, d', s)) \vee$$
$$(a = ls(d) \wedge indir(f, d)) \vee$$
$$Kindir(f, d, s) \wedge \neg((\exists d').a = mv(f, d, d') \wedge d \neq d'),$$
$$K\neg indir(f, d, do(a, s)) \equiv$$
$$((\exists d').a = mv(f, d, d') \wedge d \neq d') \vee$$
$$(a = ls(d) \wedge \neg indir(f, d)) \vee K\neg indir(f, d, s) \wedge$$
$$\neg((\exists d').a = mv(f, d', d) \wedge d \neq d' \wedge Kindir(f, d', s)).$$

$$Kreadable(f, do(a, s)) \equiv$$
$$((\exists d).a = ls(d) \wedge indir(f, d, s) \wedge readable(f, s)) \vee$$
$$Kreadable(f, s),$$
$$K\neg readable(f, do(a, s)) \equiv$$
$$((\exists d).a = ls(d) \wedge indir(f, d, s) \wedge \neg readable(f, s)) \vee$$
$$K\neg readable(f, s).$$

In the translation, the knowledge-producing effects of *ls* are distributed from the K successor state axiom into the appropriate DP successor knowledge state axioms: the universal effect into *Kindir*, $K\neg$*indir* and the conditional effect into *Kreadable*, $K\neg$*readable*. The explicit universal quantification in the K axiom is now expressed implicitly in the knowledge successor state axioms. The successor state axiom for *indir* is converted to its knowledge fluent version and also included in the axioms for *Kindir*, $K\neg$*indir*.

We must also ensure that we have an initial knowledge equivalence for fluent literals. For instance, suppose nothing is known initially about the location of a file *kr.tex*. We have the following SL and DP axioms:

$$(\forall d).\neg\mathbf{Knows}_{SL}(\mathit{indir}(\mathit{kr.tex}, d, \mathit{now}), S_0) \wedge \neg\mathbf{Knows}_{SL}(\neg\mathit{indir}(\mathit{kr.tex}, d, \mathit{now}), S_0),$$
$$(\forall d).\neg\mathit{Kindir}(\mathit{kr.tex}, d, S_0) \wedge \neg K\neg\mathit{indir}(\mathit{kr.tex}, d, S_0).$$

Using the DP theory we can now reason about knowledge change as updates to the knowledge fluents. For example, consider the situation $S_1 = \mathit{do}(\mathit{mv}(\mathit{kr.tex}, \mathit{tmp}, \mathit{papers}), S_0)$. By the successor state axiom for $K\neg$*indir*, it will be the case that $K\neg\mathit{indir}(\mathit{kr.tex}, \mathit{tmp}, S_1)$ holds. However, it will also be the case that $\neg\mathit{Kindir}(\mathit{kr.tex}, \mathit{papers}, S_1)$ and $\neg K\neg\mathit{indir}(\mathit{kr.tex}, \mathit{papers}, S_1)$ hold since initially it is not known whether *kr.tex* is in directory *tmp*. If we then consider the situation $S_2 = \mathit{do}(\mathit{ls}(\mathit{papers}), S_1)$ then either $\mathit{Kindir}(\mathit{kr.tex}, \mathit{papers}, S_2)$ or $K\neg\mathit{indir}(\mathit{kr.tex}, \mathit{papers}, S_2)$ will hold (i.e., the agent will know whether *kr.tex* is in directory *papers*), depending on whether *kr.tex* is actually in directory *papers* or not. If $\mathit{Kindir}(\mathit{kr.tex}, \mathit{papers}, S_1)$ holds, then either $\mathit{Kreadable}(\mathit{kr.tex}, S_1)$ or $K\neg\mathit{readable}(\mathit{kr.tex}, S_1)$ will also hold (i.e., the agent will know whether *kr.tex* is readable).

Moreover, our equivalence results ensure that the DP knowledge fluents can also be understood in terms of the SL theory. For instance, by Theorem 3 we will have that $\mathbf{Knows}_{SL}(\neg\mathit{indir}(\mathit{kr.tex}, \mathit{tmp}, \mathit{now}), S_1)$ holds. Also, both $\neg\mathbf{Knows}_{SL}(\mathit{indir}(\mathit{kr.tex}, \mathit{papers}, \mathit{now}), S_1)$, and $\neg\mathbf{Knows}_{SL}(\neg\mathit{indir}(\mathit{kr.tex}, \mathit{papers}, \mathit{now}), S_1)$ will hold.

6 DISCUSSION

In this paper we provide a means of translating certain types of SL theories into corresponding DP theories that avoid the use of possible worlds. As a result, reasoning about knowledge change reduces to reasoning about ordinary fluent change. With n atomic formulae, determining the truth of a formula reduces from checking 2^n possible worlds to checking the truth of $3n$ fluents in the worst case. Moreover, we can make use of standard tools such as regression for addressing issues like the projection problem (Demolombe and Pozos Parra, 2000; Reiter, 2001a). From a practical standpoint, we believe our approach will lead to more efficient implementations of systems for high-level agent control or planning. Furthermore, we believe that the tradeoffs in expressiveness do not detract from the advantages of modelling certain types of problems at the knowledge level instead of the possible world level. Indeed, recent results in knowledge-based planning (Bacchus and Petrick, 2002) lend support to the viability of such an approach.

Our results can also be extended in a number of ways. Even with our current restrictions, we are still able to model powerful (and interesting) types of sensing, such as actions with universal sensory effects or a form of conditional sensing that allows fluents to be sensed, contingent on the truth of other fluents. We are exploring extensions to our combined action theories to model more comprehensive classes of sensing. For instance, our strong restrictions on disjunctive knowledge should allow us to extend our sensing to more general formulae. Likewise, a much more expressive class of physical effects could be modelled in our representation by considering less restrictive forms of quantification in successor state axioms. Such an addition, however, will require a strengthening of our disjunctive knowledge restriction, in particular, to include knowledge of existentially quantified formulae.

We also seek to extend our knowledge equivalence results to formulae with unrestricted quantification. This would allow us equate knowledge of formulae containing K or KP (i.e., introspective formulae), currently restricted by our representation. The techniques of (Levesque, 1998), including the normal form proposed by Levesque, could be adapted for this purpose. We are also looking at the possibility of modelling knowledge reducing actions such as *forget* (see Section 4.5) in our combined action theories to take advantage of the flexibility of the DP approach and to extend our correspondence with it. An interesting discussion of some of the issues concerned with "forgetting" is presented in (Lin and Reiter, 1994).

We are also able to relax some of our assumptions. We have ignored any discussion of action preconditions, however, a simple extension to allow knowledge-based action preconditions, for instance, could be made. We could also drop our restriction that ordinary actions be distinct from knowledge-producing actions, allowing actions to have both physical and sensory effects. Finally, we are also investigating the addition of functional fluents to the representation. These and other related issues will be discussed further in (Petrick, 2003).

References

Bacchus, F. and Petrick, R. (1998). Modeling an agent's incomplete knowledge during planning and execution. In Cohn, A. G., Schubert, L., and Shapiro, S. C., editors, *Proc. of KR-98*, pages 432–443. Morgan Kaufmann Publishers.

Bacchus, F. and Petrick, R. P. A. (2002). A knowledge-based approach to planning with incomplete information and sensing. In *Proc. of AIPS-2002*. To appear.

Baral, C. and Son, T. C. (1997). Approximate reasoning about actions in presence of sensing and incomplete information. In Małuszyński, J., editor, *Proceedings of the International Symposium on Logic Programming (ILPS-97)*, pages 387–404. MIT Press.

Demolombe, R. and Pozos Parra, M. P. (2000). A simple and tractable extension of situation calculus to epistemic logic. In *Proceedings of the Twelfth International Symposium on Methodologies for Intelligent Systems (ISMIS-2000)*, pages 515–524.

Etzioni, O., Golden, K., and Weld, D. (1997). Sound and efficient closed-world reasoning for planning. *Artificial Intelligence*, 89(1–2):113–148.

Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208.

Funge, J. (1998). Interval-valued epistemic fluents. In *AAAI 1998 Fall Symposium on Cognitive Robotics*, pages 23–25, Orlando, FL.

Hintikka, J. (1962). *Knowledge and Belief*. Cornell University Press, Ithaca, New York.

Levesque, H. J. (1998). A completeness result for reasoning with incomplete first-order knowledge bases. In Cohn, A. G., Schubert, L., and Shapiro, S. C., editors, *Proc. of KR-98*, pages 14–23. Morgan Kaufmann Publishers.

Lin, F. and Reiter, R. (1994). Forget it! In Greiner, R. and Subramanian, D., editors, *Working Notes of AAAI Fall Symposium on Relevance*, pages 154–159, Menlo Park, CA, USA. AAAI.

Lin, F. and Reiter, R. (1997). How to progress a database. *Artificial Intelligence*, 92(1–2):131–167.

McCarthy, J. and Hayes, P. J. (1969). Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502.

Moore, R. C. (1985). A formal theory of knowledge and action. In Hobbs, J. R. and Moore, R. C., editors, *Formal Theories of the Commonsense World*, pages 319–358. Ablex Publishing.

Peot, M. and Smith, D. (1992). Conditional nonlinear planning. In Hendler, J., editor, *Proceedings of the First International Conference on AI Planning Systems*, pages 189–197. Morgan Kaufmann.

Petrick, R. P. A. (2003). *Modelling knowledge and sensing for effective reasoning*. PhD thesis, Department of Computer Science, University of Toronto. Forthcoming.

Pryor, L. and Collins, G. (1996). Planning for contingencies: A decision-based approach. *Journal of Artificial Intelligence Research*, 4:287–339.

Reiter, R. (2001a). *Knowledge In Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press.

Reiter, R. (2001b). On knowledge-based programming with sensing in the situation calculus. *ACM Transactions on Computational Logic*, 2(4):433–457.

Scherl, R. B. and Levesque, H. J. (1993). The frame problem and knowledge-producing actions. In *Proc. of AAAI-93*, pages 689–695. AAAI Press/MIT Press.

Shapiro, S., Lespérance, Y., Levesque, H. J., and Pagnucco, M. (2000). Iterated belief change in the situation calculus. In Cohn, A. G., Giunchiglia, F., and Selman, B., editors, *Proc. of KR-2000*, pages 527–538. Morgan Kaufmann Publishers.

Thielscher, M. (2000). Representing the knowledge of a robot. In Cohn, A. G., Giunchiglia, F., and Selman, B., editors, *Proc. of KR-2000*, pages 109–120. Morgan Kaufmann Publishers.

Weld, D. S., Anderson, C. R., and Smith, D. E. (1998). Extending graphplan to handle uncertainty & sensing actions. In *Proc. of AAAI-98*, pages 897–904, Menlo Park. AAAI Press.

Projection in Decomposed Situation Calculus

Eyal Amir
Computer Science Division
University of California at Berkeley
Soda Hall 387, Berkeley, CA 94720-1776, USA
eyal@cs.berkeley.edu

Abstract

We investigate the impact of decomposition on projection in the situation calculus. We show that performing projection with situation calculus theories can benefit from their decomposition into parts associated with sub-domains. Particularly, we provide message-passing algorithms that take advantage of the particular structure of situation calculus theories to perform the task of projection. These algorithms are shown to be sound and complete for this task for different scenarios, including actions with non-deterministic effects, partially specified initial situation and observations in situations later than the first one. They can be used for distributed reasoning about situation calculus theories or to speed up computation, in those cases where they are efficient. We characterize the kind of messages that must be sent between partitions for each of our algorithms and scenarios. This allows us to provide computational complexity results for the proposed algorithms under some assumptions. Our results are important for analyzing and devising planning, diagnosis and control algorithms for large domains that are made of interacting parts.

1 Introduction

Situation calculus (McCarthy and Hayes, 1969) is one of the leading logical formalizations for the representation of actions and change. It is used to specify high-level programs for robots (e.g., (Levesque et al., 1997)) and to do projection (e.g., (Reiter, 1992)), planning (e.g., (Green, 1969; Finzi et al., 2000)) and diagnosis (e.g., (McIlraith, 1997)) in dynamic systems. It is particularly useful for these tasks because its language is highly expressive and many extensions can be represented within it with relative ease (see (Reiter, 2001)). It also allows us to formally examine many other algorithms that reason about dynamic systems, analyze them and generalize them (e.g., (Lin and Reiter, 1995; Santibaez, 1999)). In general, reasoning about dynamic systems is computationally expensive (e.g., (Bylander, 1994; Baral et al., 2000)), whether done using situation calculus or otherwise. However, in the last 15 years some approaches that advocate decomposition of problems have been developed with some success (e.g., (Pearl, 1988; Dechter and Pearl, 1989; Darwiche, 1998; Amir and McIlraith, 2000; Pfeffer, 2001)).

In this paper we investigate the applicability of decomposition to projection in the situation calculus, projection being the prototypical reasoning problem for dynamic systems. We propose reasoning procedures for situation calculus theories that are composed of interacting sub-domains (we refer to each situation calculus theory that is associated with these sub-domains as a *partition*). Our algorithms use local computation for each partition and send messages of restricted form and length between the partitions. We prove the soundness and completeness of our algorithms, and provide computational analysis of these algorithms under different assumptions. Particularly, we provide a theorem that relates the structure of a given sequence of actions with the computational cost of projection of this sequence with our algorithms.

Our characterization of the messages that must be sent between partitions is important because it allows us to develop specialized reasoning procedures that look for these formulae directly, resulting in an efficient way to perform projection and planning in problems that are composed of related parts. Our theorems are applicable to a wide range of situation calculus theories, such as theories with non-deterministic actions, observations, knowledge-producing actions and other extensions developed for simple situation calculus theories (see (Reiter, 2001)). Finally, the results here are immediately applicable to the object-oriented situation calculus theories proposed in (Amir, 2000).

(Amir and McIlraith, 2000) provided message-passing algorithms for reasoning with logical theories that are made of interacting subtheories in the style of (Pearl, 1988). These algorithms are applicable in our setup, but for general First-Order Logic (FOL) theories they may end up producing arbitrarily many messages that may be very large. Some of our algorithms here can be seen as a restriction on the messages that these algorithms can be send between partitions.

Some proofs are omitted here for lack of space. They appear in (Amir, 2002b).

2 Background: Decomposed Situation Calculus

2.1 Situation Calculus

We review situation calculus briefly, and the reader is referred to (Reiter, 2001) for further background and details.

Situation calculus (McCarthy and Hayes, 1969) is a logical formalism for representing temporal information. The language consists of four sorts: *situations*, for situations in the world; *actions*, for events and actions; *fluents*, for situation-dependent properties; and *objects*, for simple other FOL objects.

Unless otherwise mentioned, s, a, f (or subscripted versions thereof) are variables for situations, actions and fluents, respectively. All other variables are of sort "object". The predicate $Holds(f, s)$ asserts that a fluent f holds in the situation s ($f(s)$ is used as a shorthand for $Holds(f, s)$ when such a shorthand causes no confusion). The function $result(a, s)$ returns the situation that results from performing a in s (we use $res(a, s)$ as a shorthand). $S0$ is a situation constant. Figure 1 displays A_{bw}, a sample situation calculus theory for the blocks-world domain. Capitalized symbols are constants. Free variables in axioms are universally quantified with maximum scope.

$$on(A, B, S0) \wedge on(B, Table, S0) \wedge on(C, Table, S0)$$

$$on(x, y, s) \Rightarrow above(x, y, s)$$
$$above(x, y, s) \wedge above(y, z, s) \Rightarrow above(x, z, s)$$
$$clear(b, s) \iff \forall b' \neg on(b', b, s)$$
$$handEmpty(s) \iff \forall b \neg inHand(b, s)$$
$$on(x, y, s) \Rightarrow \neg inHand(x, s) \wedge \neg inHand(y, s)$$
$$inHand(x, s) \Rightarrow \neg on(x, y, s) \wedge \neg on(y, x, s)$$
$$on(x, y, s) \wedge y \neq z \Rightarrow \neg on(x, z, s)$$

$$clear(b, s) \wedge handEmpty(s) \Rightarrow inHand(b, res(pickUp(b), s)))$$
$$(y = Table \vee clear(y, s)) \wedge inHand(x, s) \Rightarrow on(x, y, res(putOn(x, y), s))$$

Figure 1: Blocks-world in the situation calculus: A_{bw}.

In situation calculus theories, *effect axioms* are used to derive the consequences of actions. In Figure 1, effect axioms are given for the actions $putOn(x, y)$, $pickUp(b)$. *state constraints* are sentences that mention no action term. This ontology received a suitable set of foundational axioms, regarding its structure of time in (Lin and Reiter, 1994; Pinto, 1994) and others.

The *frame problem* concerns the conclusion of non-effects of actions from the known effects in a concise, correct, expressive and elaboration-tolerant manner (see (Shanahan, 1997)). One of the solutions generates *explanation closure* axioms (e.g., (Haas, 1987; Pednault, 1989; Schubert, 1990)) from the effect axioms and the domain constraints. When it is possible, we mechanically (outside the logic) add axioms saying that if something has changed, then one of the enumerated actions occurred and their proper preconditions held. For example, one axiom that is generated for explanation closure for $inHand$ in A_{bw} is $\neg inHand(x, s) \wedge inHand(x, res(a, s)) \Rightarrow (clear(x, s) \wedge handEmpty(s) \wedge a = pickUp(x))$.

(Reiter, 1991) summarized the effort and showed how to generate such axioms automatically if there are no state constraints, (Lin and Reiter, 1994) extended this process for the presence of state constraints, using deduction, and (McIlraith, 2000) gave a closed-form solution in the presence of some restricted state constraints. These solutions also add other axiom sets, including unique names axioms (UNA) for sort "actions", preconditions for executing actions (summarized by the predicate $Poss(a, s)$) and foundational axioms for situations.

All of the results in the rest of this paper are stated for situation calculus theories in which the frame problem is solved using Reiter's solution (see (Reiter, 2001)) prior to reasoning. We assume that there are no state constraints or that whatever state constraints there are were compiled into that solution by the methods of (Lin and Reiter, 1994; Pinto, 1999; McIlraith, 2000).

2.2 Combining Action Theories

The way situation calculus can be joined using object-oriented design tools was examined in (Amir, 2000). It showed that domain theories that are represented using situation calculus can be joined without the need for significant recomputation of the solution to the frame problem. We use the results mentioned in (Amir, 2000) while avoiding the use of object-oriented notation, thus sidestepping unnecessary definitions.

Consider a domain theory regarding buying and selling items. In this theory, buying an item decreases the amount of money a robot has, but also places the purchased item in the robot's hand. Selling a block has the opposite effect.

Figure 2 presents the new axioms. The blocks-world theory and the buy-sell theory share only $S0, res, Holds$ and $inHand$. Figure 3 is a diagram of the situation.

$$money(S0) = 10$$

$$money(s) \geq 0$$
$$inHand(b, s) \Rightarrow HasItem(b, s)$$

$$inHand(b, s) \wedge value(b, s) = v \wedge money(s) = m \Rightarrow$$
$$\neg hasItem(b, res(sell(b), s)) \wedge money(res(sell(b), s)) = m+v$$
$$value(b, s) = v \wedge money(s) = m \wedge m \geq v \Rightarrow$$
$$money(res(buy(b), s)) = m-v \wedge inHand(b, res(buy(b), s))$$

Figure 2: The domain of buying and selling items, $\mathcal{A}_{bs}$.

Following (Amir and McIlraith, 2000), we say that $\{\mathcal{A}_i\}_{i \leq n}$ is a *partitioning* of a logical theory $\mathcal{A}$ if $\mathcal{A} = \bigcup_i \mathcal{A}_i$. Each individual $\mathcal{A}_i$ is a set of axioms called a *partition*, $L(\mathcal{A}_i)$ is its signature (the set of non-logical symbols), and $\mathcal{L}(\mathcal{A}_i)$ is its language (the set of formulae built with $L(\mathcal{A}_i)$). The partitions may share literals and axioms.

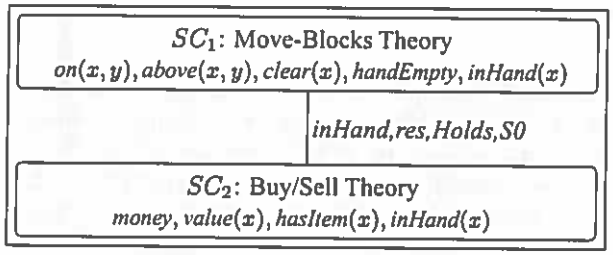


Figure 3: Combining blocks world with purchase and sale.

We call the graph in Figure 3 the *intersection graph* of the partitioned theory $\mathcal{A}_{bw} \cup \mathcal{A}_{bs}$. More generally, every partitioning of a theory induces a graphical representation, $G = (V, E, l)$, which we call the partitioning's *intersection graph*. Each node of the intersection graph, 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 non-logical symbol in common $(E = \{(i,j) \mid L(\mathcal{A}_i) \cap L(\mathcal{A}_j) \neq \emptyset\})$, and the edges are labeled with the set of symbols that the associated partitions share $(l(i,j) = L(\mathcal{A}_i) \cap L(\mathcal{A}_j))$. 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, $l(i,j) = \emptyset$.

(Amir, 2000) showed that adding the explanation-closure solution to the frame problem can be done in a way that ensures that the graph does not become more connected. Situation calculus theories that are made of such connected subtheories (two subtheories or more) are called *oo-sitcalc theories*. They can be built from component theories or be the result of manual or automatic decomposition (Amir and McIlraith, 2000; Amir, 2001).

2.3 Message-Passing

Figure 4 displays MESSAGE-PASSING (MP), a message-passing algorithm proposed in (Amir and McIlraith, 2000) for partition-based logical reasoning. It takes as input a partitioned theory, $\mathcal{A}$, an associated graph structure $G = (V, E, l)$, and a query formula Q in $\mathcal{L}(\mathcal{A}_k)$, and returns YES if the query was entailed by $\mathcal{A}$. The algorithm uses procedures that generate consequences (consequence finders) as the local reasoning mechanism within each partition or graphical node. It passes a concluded formula to an adjacent node if the formula's signature is in the communication language l of the adjacent node, and that node is on the path to the node containing the query.

The messages in this algorithm are sent in a single direction, the direction of the *goal partition*. To determine the direction in which messages should be sent in the graph G, step 1 in MP computes a strict partial order over nodes in the graph using the partitioning together with a query, Q.

Definition 2.1 ($\prec$) *Given partitioned theory* $\mathcal{A} = \bigcup_{i \leq n} \mathcal{A}_i$, *associated graph* $G = (V, E, l)$ *and query* $Q \in \mathcal{L}(\mathcal{A}_k)$, *let* $dist(i,j)$ $(i, j \in V)$ *be the length of the shortest path between nodes* i, j *in* G. *Then* $i \prec j$ *iff* $dist(i,k) < dist(j,k)$.

PROCEDURE MESSAGE-PASSING (MP)($\{\mathcal{A}_i\}_{i \leq n}, G, Q$)
$\{\mathcal{A}_i\}_{i \leq n}$ a partitioning of the theory $\mathcal{A}$, $G = (V, E, l)$ a graph describing the connections between the partitions, Q a query in $\mathcal{L}(\mathcal{A}_k)$ $(k \leq n)$.

1. Determine $\prec$ as in Definition 2.1.
2. Concurrently,
   (a) Perform consequence finding for each of the partitions $\mathcal{A}_i$, $i \leq n$.
   (b) For every $(i,j) \in E$ such that $i \prec j$, for every consequence φ of $\mathcal{A}_j$ found (or φ in $\mathcal{A}_j$), if $\varphi \in \mathcal{L}(l(i,j))$, then add φ to the set of axioms of $\mathcal{A}_i$.
   (c) If Q is proven[a] in $\mathcal{A}_k$, return YES.

[a] Derive a subsuming formula or initially add $\neg Q$ to $\mathcal{A}_k$ and derive inconsistency.

Figure 4: A forward message-passing algorithm (McIlraith and Amir, 2001).

It was shown in (Amir and McIlraith, 2000; McIlraith and Amir, 2001) that this kind of algorithm is complete and sound for partitioned theories, and that it sometimes improves running time compared to standard deduction methods.

Completeness of the algorithm assumes that the graph G used in the algorithm is *properly labeled* (in other fields this is sometimes called *satisfies the running intersection property*). The condition of proper labeling for a graph $G(V, E, l)$ was defined in (McIlraith and Amir, 2001) to say that G is a tree of partitions such that 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 $l(i, j) \supseteq L(\mathcal{B}_1) \cap L(\mathcal{B}_2)$. This condition always holds if the intersection graph of a partitioned theory is a tree, and there are algorithms for converting any graph into such a tree, e.g., BREAK-CYCLES(G) in (Amir and McIlraith, 2000). The *width* of a properly-labeled tree is the size of the largest partition (size here includes the number of fluents in the partition and on the labels of the partition's edges). The *treewidth* of SC is the lowest width among all properly-labeled trees for SC.

MP is immediately applicable to both projection and planning in situation calculus if we partition those theories as in the previous section. However, the number of messages sent and their lengths can be too large. The rest of the paper characterizes the messages that must be sent in similar algorithms and in bi-directional MP algorithms, showing that we can restrict the messages significantly for the task of projection.

3 Projection Using Bi-Directional Passing of Restricted Messages

In this section we propose two message-passing algorithms that perform projection in situation calculus theories by sending only messages of a restricted-form. Both algorithms perform projection by sending messages back-and-forth between partitions. They differ in the restrictions they put on the messages and in the range of theories for which they are applicable.

In the following we assume that each observation that we have (about $S0$ or otherwise, if allowed in our scenario) can be represented using the vocabulary that we associate with a single partition (or can be compiled (e.g., using the methods of (McIlraith and Amir, 2001)) into equivalent multiple observations that satisfy this condition).

Also, for simplicity we assume that fluents are propositional (i.e., there are no function fluents or predicate fluents, but rather fluents are named and no function generates new fluents from others).

3.1 Deterministic Situation Calculus Theories

Our first algorithm sends messages of the form $Holds(f, S_j)$ between partitions, for f and S_j being ground terms of sort fluent and situation, respectively. This algorithm is applicable to situation calculus theories that have a fully specified initial situation and deterministic actions. The procedure is present in Figure 5.

PROCEDURE MP-2dir-det-SitCalc (MP2d)($\{SC_i\}_{i\leq n}, G, Q$)

$\{SC_i\}_{i\leq n}$ an oo-sitcalc theory, $G = (V, E, l)$ a graph describing the connections between the partitions, Q a query in $\mathcal{L}(SC_k)$ ($k \leq n$).

1. Concurrently,
   (a) Perform consequence finding for each of the partitions SC_i, $i \leq n$.
   (b) For every $(i, j) \in E$, for every consequence $Holds(\varphi, S)$ of SC_j found (or $Holds(\varphi, S)$ in SC_j), if $\varphi \in \mathcal{L}(l(i, j))$ is a fluent term and S a ground situation term, then add $Holds(\varphi, S)$ to the set of axioms of SC_i.
   (c) If Q is proven[a] in SC_k, return YES.

[a] Derive a subsuming formula or initially add $\neg Q$ to SC_k and derive inconsistency.

Figure 5: A bi-directional message-passing algorithm for deterministic scenarios.

Theorem 3.1 (Soundness and Completeness of MP2d) *Let $SC = \{SC_i\}_{i\leq n}$ be an oo-sitcalc theory, $a_1, \ldots, a_m$ actions in $L(SC)$ and for all $j \leq m$ $S_j = result(a_j, res(a_{j-1}, ..., res(a_1, S0)))$. Let $\varphi \in \mathcal{L}(SC_k)$ be a fluent term, for $k \leq n$. Then $SC \models Holds(\varphi, S_j)$ iff MP2d outputs YES for the query $Holds(\varphi, S_j)$ stated in partition SC_k.*

Most of the rest of this section is devoted to proving this theorem. The proof outline is as follows: We *unroll* the *result* function for a partitioned situation calculus theory, SC, that has only two partitions, SC_1 and SC_2 (see Figure 6). We show that the resulting theory, SC', is equivalent to the first one for a set of queries. We use this and the soundness and completeness of MP for arbitrary partitioned theories to find the messages that can be sent in MP running on SC'. These messages are then translated back to the original system, resulting in the desired characterization. Finally, the generalization of this characterization to the case of a tree of partitions (instead of only two partitions) is done by induction.

Definition 3.2 (Unrolling *result*) *Let $SC = \{SC_1, SC_2\}$ be an oo-sitcalc theory with two partitions (e.g., see Figure 3). We say that SC'* unrolls *SC for the action sequence $a_1, \ldots, a_n$ if $SC' = \{SC^i\}_{1\leq i\leq n}$ and for every $i \leq n$ SC^i is a copy of SC_1 (if a_i resides[1] in SC_1) or SC_2 (if a_i resides in SC_2) that we change in three ways: First, we*

[1] We say that a_i *resides* in SC_j if the effect axioms of a_i are in SC_j.

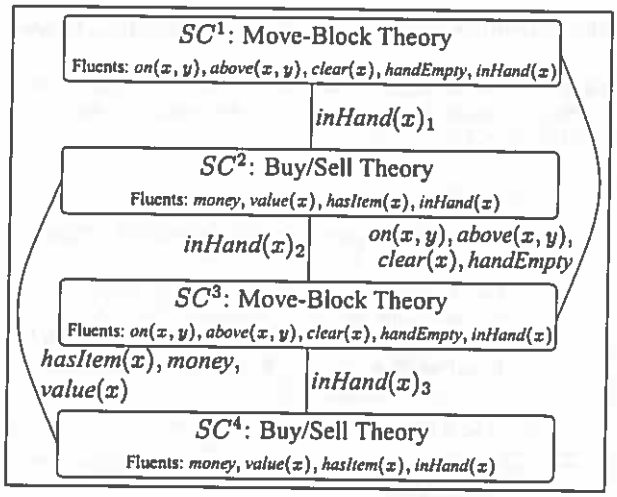


Figure 6: High-level diagrammatic view of unrolling *result*.

replace every nonlogical symbol X that is not a fluent name by a new symbol $SC^i.X$. Then, we add new propositional symbols f_i for every fluent name $f \in L(SC^i)$. Finally, for every fluent name $f \in L(SC^{i-1}) \cap L(SC^i)$ we add the axioms

$$f_{i-1} \iff SC^i.Holds(f, SC^{i-1}.res(SC^i.a_{i-1}, ..., res(SC^i.a_1, SC^i.S0)...))$$
$$f_i \iff SC^i.Holds(f, SC^i.res(SC^i.a_i, ..., res(SC^i.a_1, SC^i.S0)...))$$

and for every $f \in L(SC_i) \setminus L(SC_{i-1})$ we add the axioms

$$f_j \iff f_{j+1} \ ..., \ f_{i-2} \iff f_{i-1}$$

if $j < i$ such that SC^i, SC^j are copies of the same partition (SC_1 or SC_2) and j is the largest such index.

Figure 6 is an example unrolling of the theory $SC = \{\mathcal{A}_{bw}, \mathcal{A}_{bs}\}$ for four actions. The intuition behind this unrolling is that we try to keep the partitions $SC_1 = \mathcal{A}_{bw}, SC_2 = \mathcal{A}_{bs}$ as separate as possible. Every partition of axioms in Figure 6 corresponds to a situation, action and the action's result in that situation. It is important that every partition SC^i does not include the entire state, nor does it need to reason about the absent fluents.

The definition roughly suggests that partition SC^i receives information from the previous partition and one more partition, if its immediate predecessor is not a copy of the same original partition of SC. After applying the action a_i in SC^i we send some of the fluents values in the resulting situation to SC^{i+1} and the rest of the fluents of $L(SC^i)$ to the next partition down the chain that needs them (a partition that is a copy of the same original partition of SC as SC^i).

We show that this *unrolling* is equivalent to the original structure for queries of the form $Holds(\varphi, res(a_n, res(..., res(a_1, S0))))$.

Lemma 3.3 (Equivalence of Unrolling) *Let $SC = \{SC_1, SC_2\}$ be an oo-sitcalc theory (as in Figure 3), and let SC' be the unrolling of SC for $a_1, ..., a_n$. Let $S_i = result(a_i, res(a_{i-1}, ..., res(a_1, S0)))$ for all $i \leq n$, and let $\varphi \in \mathcal{L}(SC^n)$ a fluent term. Then,*

$$SC \models Holds(\varphi, S_n) \iff SC' \models SC^n.Holds(\varphi, res(SC^n.a_n, SC^n.S0)).$$

PROOF See Appendix A.1. ■

For the following lemma we assume full knowledge of the state $S0$ and actions that are deterministic.

Lemma 3.4 (MP in Unrolled Theory) *Let $SC = \{SC_1, SC_2\}$ be an oo-sitcalc theory, $a_1, ..., a_n$ actions in $L(SC)$ and $S_i = res(a_i, res(a_{i-1}, ..., res(a_1, S0)))$ for all $i \leq n$. Let $\varphi \in \mathcal{L}(SC_j)$ be a fluent term, for $j \in \{1, 2\}$. Then $SC \models Holds(\varphi, S_n)$ iff MP outputs YES for the query[2] $SC^n.Holds(\varphi, SC^n.S_n)$ stated in partition SC^n.*

It is important to notice that we do not convert the graph G into a tree before running MP. The fact that this lemma holds even without running BREAK-CYCLES on G is a result of our assumption about deterministic actions and full knowledge of the fluents in $S0$.

Now we can show that the only messages that need to be sent in a back-and-forth Message-Passing are of the form of a single state observation or constraint. For the following theorem we assume full knowledge of the state $S0$ and actions that are deterministic.

Theorem 3.5 (MP2d is Complete & Sound for 2 Parts.) *Let $SC = \{SC_1, SC_2\}$ be an oo-sitcalc theory (as in Figure 3), $a_1, ..., a_n$ actions in $L(SC)$ and $S_i = result(a_i, res(a_{i-1}, ..., res(a_1, S0)))$ for all $i \leq n$. Let $\varphi \in \mathcal{L}(SC_j)$ a fluent term, for $j \in \{1, 2\}$. Then $SC \models Holds(\varphi, S_n)$ iff MP2d outputs YES for the query $Holds(\varphi, S_n)$ stated in partition SC_j.*

PROOF SKETCH From Lemma 3.3 we know that $SC \models Holds(\varphi, S_G)$ iff $SC' \models SC^n.Holds(\varphi, SC^n.S_n))$. Furthermore, from Lemma 3.4 we know that the messages that need to be passed between partitions that are copies of SC^1 and those that are copies of SC^2 are of a form equivalent to $Holds(\psi, S_i)$. ■

The generalization of the last theorem to the tree case is done by induction, leading to the conclusion of the proof of Theorem 3.1. ■

[2] We abuse notations here and write $SC^n.S_n$ for $SC^n.res(SC^n.a_n, ..., SC^n.res(SC^n.a_1, SC^n.S0)...)$.

For this special case (deterministic actions and fully known initial state) we can show that MP2d is complete even if G is not a tree. This follows from a similar argument that we do not bring here.

3.2 Non-Deterministic Situation Calculus Theories

It is important to notice that Theorem 3.1 does not hold if $S0$ is not fully specified or some actions have non-deterministic effects. To see this, assume that we know that $Holds(on(A,B) \vee inHand(A), S0)$. This implies that

$$Holds(on(A,B) \vee inHand(D), res(buy(D), res(sell(A), S0)))$$

is a valid consequence of $\mathcal{A}_{bw}$. However, we cannot prove this using messages of the form $Holds(\psi, S_i)$ when ψ is a fluent term based on *inHand* because there is no conclusion that we can draw about $inHand(A)$ or $inHand(D)$ or any relationship between them in any single situation. We can conclude

$$Holds(inHand(A), S0) \Rightarrow Holds(inHand(D), res(buy(D), res(sell(A), S0))),$$

but this is a formula that includes two different situations.

Nonetheless, a generalization of Theorem 3.1 holds for the case of nondeterministic actions, partially-specified initial state and possible observations about states later than $S0$. By nondeterministic actions we refer to actions whose effect axioms specify results that are not a single conjunction of fluents. The specification of nondeterministic actions and the solution to the frame problem in such settings take different semantics in different works (Lin, 1996; Levesque et al., 1997). Here, we assume that the solution is given using some added first-order axioms, but we do not assume any particular information about them. Similar results can be achieved for the GOLOG model of nondeterministic selection of actions (Levesque et al., 1997).

Our algorithm performs projection by sending messages of the form $Holds(\psi_1, S_{i_1}) \wedge \ldots \wedge Holds(\psi_a, S_{i_a}) \Rightarrow Holds(\psi_{a+1}, S_{i_{a+1}})$ between partitions, for ψ_j being ground terms of sort fluent and S_{i_j} being ground terms of sort situation. It is more generally applicable than the first algorithm, allowing theories that include nondeterministic effects of actions, partially specified first situation and observations about later situations. The procedure is present in Figure 7.

Theorem 3.6 (Soundness and Completeness of MP2n) *Let $SC = \{SC_i\}_{i \leq n}$ be an oo-sitcalc theory, $a_1, \ldots, a_m$ actions in $L(SC)$ and for all $j \leq m$ $S_j = result(a_j, res(a_{j-1}, \ldots, res(a_1, S0)))$. Let $\varphi \in \mathcal{L}(SC_k)$ be a fluent term, for $k \leq n$. Then $SC \models Holds(\varphi, S_j)$ iff MP2n outputs YES for the query $Holds(\varphi, S_j)$ stated in partition SC_k.*

PROCEDURE MP2-nondet-SitCalc (MP2n)($\{SC_i\}_{i \leq n}, G, Q$)

$\{SC_i\}_{i \leq n}$ an oo-sitcalc theory, $G = (V, E, l)$ a graph describing the connections between the partitions, Q a query in $\mathcal{L}(SC_k)$ ($k \leq n$).

1. Concurrently,
   (a) Perform consequence finding for each of the partitions SC_i, $i \leq n$.
   (b) For every $(i,j) \in E$, for every consequence of the form $\Psi = Holds(\psi_1, S_{i_1}) \wedge \ldots \wedge Holds(\psi_a, S_{i_a}) \Rightarrow Holds(\psi_{a+1}, S_{i_{a+1}})$ of SC_j found (or $\Psi \in SC_j$), if $\Psi \in \mathcal{L}(l(i,j))$, then add Ψ to the set of axioms of SC_i.
   (c) If Q is proven[a] in SC_k, return YES.

[a] Derive a subsuming formula or initially add $\neg Q$ to SC_k and derive inconsistency.

Figure 7: A bi-directional message-passing algorithm for nondeterministic scenarios.

The proof of this theorem follows from Lemma 3.3 and the soundness and completeness of MP for properly labeled graphs (McIlraith and Amir, 2001). The proof follows if we notice that running MP on an oo-sitcalc theory as in Figure 6 after running BREAK-CYCLES (Amir and McIlraith, 2000) results in messages that can be converted to the proper form for our theorem.

4 Projection Using Messages-Passing in One Direction

The previous section characterized those messages that must be sent in a back-and-forth message passing proof over an oo-sitcalc theory. In this section we examine the case of sending messages in only one direction, the way MP does. We find that the messages that must be sent are of a similar form to that given for nondeterministic domains in the previous section: $Holds(\psi_1, S_{i_1}) \wedge \ldots \wedge Holds(\psi_a, S_{i_a}) \Rightarrow Holds(\psi_{a+1}, S_{i_{a+1}})$. Our algorithm is shown in Figure 8.

Surprisingly, we must send the same kind of messages even if we limit our attention to deterministic domains, i.e., formulae of the form $Holds(f, S)$ or $Holds(\psi, S) \Rightarrow Holds(\psi', S')$ are not enough. Roughly speaking, the reason for this is that when we send messages in a single direction each partition has to send enough messages to account for all the possible situations to which it might have been applied if we used bi-directional message-passing.

In the following theorem we allow partially specified initial situation, nondeterministic effects of actions and observations in situations later than $S0$.

PROCEDURE MP-nondet-SitCalc(MPn)($\{SC_i\}_{i\leq n}, G, Q$)
$\{SC_i\}_{i\leq n}$ an oo-sitcalc theory, $G = (V, E, l)$ a graph describing the connections between the partitions, Q a query in $\mathcal{L}(SC_k)$ $(k \leq n)$.

1. Determine $\prec$ as in Definition 2.1.
2. Concurrently,
   (a) Perform consequence finding for each of the partitions $SC_i, i \leq n$.
   (b) For every $(i,j) \in E$ such that $i \prec j$, for every consequence of the form $\Psi = Holds(\psi_1, S_{i_1}) \wedge \ldots \wedge Holds(\psi_a, S_{i_a}) \Rightarrow Holds(\psi_{a+1}, S_{i_{a+1}})$ of SC_j found (or $\Psi \in SC_j$), if $\Psi \in \mathcal{L}(l(i,j))$, then add Ψ to SC_i's axioms set.
   (c) If Q is proven[a] in SC_k, return YES.

[a]Derive a subsuming formula or initially add $\neg Q$ to SC_k and derive inconsistency.

Figure 8: A forward message-passing algorithm for projection in deterministic and nondeterministic domains.

Theorem 4.1 (Soundness and Completeness of MPn) *Let $SC = \{SC_i\}_{i\leq n}$ be an oo-sitcalc theory, $a_1, \ldots, a_m$ actions in $L(SC)$ and for all $j \leq m$ $S_j = result(a_j, res(a_{j-1}, ..., res(a_1, S0)))$. Let $\varphi \in \mathcal{L}(SC_k)$ a fluent term, for $k \leq n$. Then $SC \models Holds(\varphi, S_j)$ iff MPn outputs YES for the query $Holds(\varphi, S_j)$ stated in partition SC_k.*

5 Structure and Computational Analysis

In situations where computation is distributed over several computers or agents, our procedures and their alike are the only mean for performing inference and projection. However, in cases where distribution is not necessary, there are simple methods that may outperform our procedures significantly, especially if we allow long messages to be sent. The computation offered by the procedures above can be very costly if the length of messages is not bounded.

In those cases, when is it better to use our procedures? In general, can we estimate the amount of computation involved given a sequence of actions and a set of partitions? In this section we characterize those cases and yield a few insights into the related problem of planning. Our analysis is based on the observation that the size of the messages can be kept small if there is only little back and forth transition between partitions in a given sequence of actions.

We make this intuition precise using the following definition.

Definition 5.1 *Let $SC = \{SC_i\}_{i\leq n}$ be an oo-sitcalc theory, $a_1, \ldots, a_m$ actions in $L(SC)$ and $G(V, E, l)$ a graph describing the connections between partitions. For $(i,j) \in E$, we say that $\langle a_{l_1}, a_{l_2}, \ldots, a_{l_k}\rangle$ $(\forall x < k\ 1 \leq l_x < l_{x+1} \leq m)$ is an* influencing sequence *for SC_i from SC_j if (A) there are $t_1, t_2 \leq m$ such that $l_k < t_1 \leq t_2$, $a_{t_1}, a_{t_2} \in \mathcal{L}(SC_i)$ and a_{t_1} has preconditions in $\mathcal{L}(l(i,j))$ and a_{t_2} has effects in $\mathcal{L}(l(i,j))$; and (B) for every $x \leq k$,*

1. *$a_{l_x} \in \mathcal{L}(SC_j)$,*
2. *a_{l_x} has effects in $\mathcal{L}(l(i,j))$, and*
3. *if $x < k$, then there is $l' \leq m$ such that $l_x < l' < l_{x+1}$, $a_{l'} \in \mathcal{L}(SC_i)$ and $a_{l'}$ has preconditions in $\mathcal{L}(l(i,j))$.*

The intuition behind this definition is that if we have a sequence of actions such that some of them belong to SC_i, SC_j, then SC_i needs to send SC_j a message of the form

> if S_1 was to be the case after a_{l_1}, and S_2 was to be the case after a_{l_2}, and ... and S_{l_k} was to be the case after a_{l_k}, then $S_{l_{k+1}}$ will be the case after $a_{l_{k+1}}$.

The length of the largest influencing sequence for SC_i from SC_j in a sequence of actions is exactly the maximal number of situations that must participate in a message that is sent from SC_i to SC_j.

Theorem 5.2 *Let $SC = \{SC_i\}_{i\leq n}$ be an oo-sitcalc theory, $a_1, \ldots, a_m$ actions in $L(SC)$ and $G(V, E, l)$ a tree describing the connections between partitions. Let SC_i, SC_j be two partitions such that $(i,j) \in E$, and let B_i, B_j be the two connected components of G that result if we remove (i,j) from E. In MP2n and in MPn, the only messages that we need to send from SC_i to SC_j are of the form*

$$Holds(\psi_1, S_{t_1}) \wedge \ldots \wedge Holds(\psi_l, S_{t_l}) \Rightarrow Holds(\psi_{l+1}, S_{t_{l+1}})$$

such that $\langle a_{t_1}, ..., a_{t_l}\rangle$ is an influencing sequence for B_i from B_j.

Thus, if there is no influencing sequence in our sequence of actions (e.g., when the sequence of actions is such that the actions are grouped into the partitions, and the order between the action-groups follows $\prec$ from Definition 2.1), then the only messages that need to be sent are of the form $Holds(\varphi, S)$. If the only influencing sequences for $a_1, \ldots, a_m$ are of length 1, then the only messages that need to be sent are of the form $Holds(\varphi, S) \Rightarrow Holds(\varphi', S')$.

Corollary 5.3 *Let $a_1, \ldots, a_m$ be a sequence of actions and let $SC = \bigcup_{i=1}^{n} SC_i$ be an oo-sitcalc theory with treewidth k_1 (treewidth is defined in Section 2.3). If the largest influencing sequence is of length k_2, then the projection with MPn takes time $O(2^{k_1 * k_2})$.*

This compares well with the coNP-completeness results for projection with partial information and nondeterministic actions (Baral et al., 2000; Amir, 2002a).

While doing projection we may sometimes take the given order of actions and change it to suit our purposes (not the actual execution, which is given, but the sequence of actions that we process in our algorithm). For example, we can try to group the actions so that they minimize $k_1 * k_2$ in Corollary 5.3 by looking at the dependencies between the actions, and rearranging the actions so that those dependencies are not altered or broken.

This result also suggests a new planning goal. When planning, try to find those plans that have the best *aggregation* of actions. Finding plans that have influencing sequences of length at most k_2, if such exist, takes time that is proportional to $2^{k_1 * k_2}$. This is subject of ongoing work.

6 Conclusions

This paper presented three novel algorithms for reasoning in partitioned domains using situation calculus. The algorithms were shown to be sound and complete for their respective classes of situation calculus theories. The first is a back-and-forth message-passing algorithm that needs to send only single-state formulae between partitions. The second is a back-and-forth message-passing algorithm that needs to send only effect-style formulae (two-state formulae) between partitions. The last algorithm is a single-direction message passing algorithm that needs to send k-state effect formulae between partitions. The first algorithm is applicable to situation calculus theories that have fully-specified initial states and deterministic effects of actions. The second and third are applicable to more general theories, including those that have only a partially specified initial state and nondeterministic effects of actions.

The results of this paper are important for several applications. First, in domains where reasoning is distributed among several machines or agents these algorithms allow us to perform projection. Also, algorithms for planning and diagnosis can be devised using our algorithms and theorems. Our results can serve as the basis for algorithms for reasoning about interacting agents and planning for collaborating agents. We can also analyze existing planning algorithms that are built around the idea of decomposition, such as (Lansky, 1988; Lansky and Getoor, 1995; Frank et al., 2000).

Also, our results can serve as a basis for reasoning algorithms for Markov Decision Processes (MDPs) and inference algorithms for some dynamic Bayes networks. They particularly shed some light on the applicability of algorithms for first-order MDPs (Boutilier et al., 2001). Finally, we are interested in building AI architectures that are based on networks of interacting interacting knowledge bases, and the algorithms and theorems offered in this paper are important for the understanding and scaling up of such architectures (see e.g., (Amir and Maynard-Reid II, 1999)).

7 Acknowledgments

I wish to thank Sheila McIlraith and Stuart Russell for discussions on subjects related to this paper. This research was supported by DARPA grant N66001-00-C-8018 (RKF program) and National Science Foundation grant ECS-9873474.

A Proofs

A.1 Lemma 3.3: Equivalence of Unrolling

Backward Direction The backward direction is seen by viewing the way MP works on SC' (after applying BREAK-CYCLES to those *bypassing* edges in the graph). Let S be a message-passing proof of $SC' \models SC^n.Holds(\varphi, SC^n.res(SC^n.a_n, SC^n.S0))$ represented as a sequence of formulae (this is the traditional Frege-Hilbert proof in which every formula is derived from previous ones in the sequence using a rule in the logic).

Every step in this proof is a possible in a regular proof in SC because every deduction step in this proof can be made between the corresponding axioms in SC. The only axiom that we add in SC' to the translated axioms of SC are the equivalence axioms $f_{i-1} \iff f_j$ for fluents in far-apart partitions. However, this axiom is a translation of a valid consequence of SC: $Holds(f, S_{i-1}) \iff Holds(f, S_j)$.

To see that this is a valid consequence, assume, without loss of generality, that SC^i is a translated copy of SC_1 (as in Definition 3.2). Then this formula follows from SC because f does not occur in $L(SC_2)$ and thus does not appear in $SC^{j+1}, ..., SC^{i-1}$. Because we do not have domain constraints (or they were compiled away) and we use explanation closure, this implies that the explanation closure axioms in SC for the fluent f show that f does not change its value during the execution of actions $a_{j+1}, ..., a_{i-1}$. Thus, $SC \models Holds(f, S_{i-1}) \iff Holds(f, S_j)$. As a result, we can translate the proof in SC' into a proof in SC, so

$$SC' \models SC^n.Holds(\varphi, SC^n.res(SC^n.a_n, SC^n.S0)) \Rightarrow SC \models Holds(\varphi, S_G).$$

Forward Direction For the forward direction we show that for every model $\mathcal{M}'$ of SC' there is a model $\mathcal{M}$ of SC such that for every $i \leq n$, for every $\psi \in \mathcal{L}(SC^i)$ a fluent term, if $\mathcal{M}' \models$

$SC^i.Holds(\psi, SC^i.res(SC^i.a_i, ..., SC^i.res(SC^i.a_1, SC^i.S0)...))$, then $\mathcal{M} \models Holds(\psi, S_i)$.

Only Effect Axioms We show this for the case of situation calculus theories in which there are only effect axioms (no explanation closure) first. We then generalize this to the case including explanation closure, domain constraints and observations (in $S0$ and otherwise).

Assume otherwise. Let i be the first index for which this assertion is not true. For $i = 0$ look at SC^1. It is isomorphic to SC_j (for $j = 1$ or $j = 2$, whichever SC^1 is a copy of) under syntactic translation. Since there are only effect axioms in our theory, there are no domain constraints or observations regarding $S0$. Thus, there are no restrictions on $S0$ in SC and there is a model $\mathcal{M}$ of SC that has the fluent values specified by $\mathcal{M}'$ for fluents $L(SC^1)$. Thus, $\mathcal{M}' \models SC^i.Holds(\psi, SC^i.res(SC^i.a_i, ..., SC^i.res(SC^i.a_1, SC^i.S0)))$ implies that $\mathcal{M} \models Holds(\psi, S_i)$.

Thus, $i > 0$ (recall that i is the first index for which our assertion is not true). Thus, the value of $Holds$ in $S_0, \ldots, S_i$ (in the respective partitions $SC^1, ..., SC^i$) according to $\mathcal{M}'$ is not consistent with the axioms of SC, but the value of $Holds$ in $S_0, \ldots, S_{i-1}$ is consistent with SC.

Let $\mathcal{A}'$ be the set of formulae of the form $SC^j.Holds(\psi, SC^j.res(SC^j.a_j, ..., SC^j.res(SC^j.a_1, SC^j.S0)))$ or $SC^j.Holds(\psi, SC^j.res(SC^j.a_{j-1}, ..., SC^j.res(SC^j.a_1, SC^j.S0)))$ that $\mathcal{M}'$ satisfies, for $j \leq i$. Let $\mathcal{A}$ be the translation of $\mathcal{A}'$ into $L(SC)$ (essentially removing SC^j from all the axioms in $\mathcal{A}'$).

$\mathcal{A}$ is consistent because the only way it may not be consistent is if it includes both $SC^j.Holds(\psi, SC^j.res(SC^j.a_{j-1}, ..., SC^J.res(SC^j.a_1, SC^j.S0)))$ and $\neg SC^{j-1}.Holds(\psi, SC^{j-1}.res(SC^{j-1}.a_{j-1}, ..., SC^{j-1}.res(SC^{j-1}.a_1, SC^{j-1}.S0)))$ for some formula $\psi \in \mathcal{L}(SC^j) \cap \mathcal{L}(SC^{j-1})$, but this is inconsistent with the set of formulae $f_{j-1} \iff SC^j.Holds(\psi, SC^j.res(SC^j.a_{j-1}, ..., SC^j.res(SC^j.a_1, SC^j.S0)))$ and $f_{j-1} \Leftrightarrow SC^{j-1}.Holds(\psi, SC^{j-1}.res(SC^{j-1}.a_{j-1}, ..., SC^{j-1}.res(SC^{j-1}.a_1, SC^{j-1}.S0)))$.

Furthermore, let $\mathcal{A}_0$ be the set of formulae of $\mathcal{A}$ that do not mention a_i. Then, $\mathcal{A}_0$ is consistent with SC, but $\mathcal{A}$ is not. Let $\bar{SC}$ be the set of axioms of SC with the additional precondition that the variable s in those axioms that mention a_i is different from S_{i-1}. $\bar{SC} \cup \mathcal{A}_0$ is consistent because $SC \cup \mathcal{A}_0$ is. Consequently, $\bar{SC} \cup \mathcal{A}$ is consistent because there are no axioms in $\bar{SC}$ that can interact with those of $\mathcal{A} \setminus \mathcal{A}_0$ (there is no effect axiom in $\bar{SC}$ that tells us anything about this particular situation).

Let $\mathcal{M}$ be a model of $\bar{SC} \cup \mathcal{A}$. We show that $\mathcal{M} \models SC$. Assume not. Then, there is an effect axiom in SC, $\Psi = \forall s\ Holds(\psi_1, s) \Rightarrow Holds(\psi_2, res(a_i, s))$, such that $\mathcal{M} \not\models \Psi$. Thus, $\mathcal{M} \models \neg\Psi$, because $\mathcal{M}$ is a structure and Ψ a closed formula. Since $\mathcal{M} \models \bar{\Psi}$ (for $\bar{\Psi}$ being the matching axiom in $\bar{SC}$) we get that $\mathcal{M} \models \neg(Holds(\psi_1, S_{i-1}) \Rightarrow Holds(\psi_2, res(a_i, S_{i-1})))$. Rewriting this formula yields $\mathcal{M} \models Holds(\psi_1, S_{i-1}) \wedge \neg Holds(\psi_2, res(a_i, S_{i-1}))$.

Let $SC^i.\Psi = \forall s\ SC^i.Holds(\psi_1, s) \Rightarrow SC^i.Holds(\psi_2, SC^i.res(SC^i.a_i, s))$. By the way we defined SC^i, the axiom $SC^i.\Psi$ appears in SC^i. In particular $SC^i \models SC^i.Holds(\psi_1, SC^i.S_{i-1}) \Rightarrow SC^i.Holds(\psi_2, SC^i.res(SC^i.a_i, SC^i.S_{i-1}))$.

From the paragraph preceding the last one, $\mathcal{M} \models Holds(\psi_1, S_{i-1})$. Furthermore, $\psi_1 \in \mathcal{L}(SC^i)$ by the definition of SC'. Since $\mathcal{M} \models Holds(\psi_1, S_{i-1})$ it must be that $\mathcal{M}' \models SC^i.Holds(\psi_1, SC^i.S_{i-1})$ because otherwise $\mathcal{M}' \models SC^i.Holds(\neg\psi_1, SC^i.S_{i-1})$ which implies $\mathcal{M} \models Holds(\neg\psi_1, S_{i-1})$, which is a contradiction.

Thus, $\mathcal{M}' \models SC^i.Holds(\psi_1, SC^i.S_{i-1})$, and we get that $\mathcal{M}' \models SC^i.Holds(\psi_2, SC^i.res(SC^i.a_i, SC^i.S_{i-1}))$. This contradicts our previous consequence that $\mathcal{M} \models \neg Holds(\psi_2, res(a_i, S_{i-1}))$ (which implies $\mathcal{M}' \models \neg SC^i.Holds(\psi_2, SC^i.res(SC^i.a_i, SC^i.S_{i-1}))$).

Thus, $\mathcal{M} \models SC \cup \mathcal{A}$ which concludes the induction step.

Adding Explanation Closure First we examine the way that we add explanation closure axioms to SC and to SC'. Without loss of generality, assume that our explanation closure axiom Ψ is of the form

$$\Psi = \forall s, a\ Holds(f, res(a, s)) \Rightarrow (Holds(\psi_1, s) \wedge a = A_1) \vee ... \vee (Holds(\psi_k, s) \wedge a = A_k) \vee (Holds(f, s) \wedge a \neq A_1 ... A_k)$$

If $f \in L(SC_1) \setminus L(SC_2)$, then all the effect axioms that influence f are in SC_1 and all of $A_1, ..., A_k$ are in $L(SC_1)$. Thus, Ψ is added to SC_1 in SC and $SC^i.\Psi$ is added to the copies SC^i of SC_1 in SC'. The situation is the opposite if $f \in L(SC_2) \setminus L(SC_1)$.

If $f \in L(SC_1) \cap L(SC_2)$, then there are effect axioms for different actions in SC_1, SC_2 that influence f. Let $A_1, ..., A_l$ be the actions for whose effect axioms are in SC_1 and $A_{l+1}, ..., A_k$ be the actions for whose effect axioms are in SC_2. In SC_1 we add the effect axiom

$$\Psi_1 = \forall s, a\ Holds(f, res(a, s)) \Rightarrow (Holds(\psi_1, s) \wedge a = A_1) \vee ... \vee (Holds(\psi_l, s) \wedge a = A_l) \vee (a = A_{l+1} ... A_k) \vee (Holds(f, s) \wedge a \neq A_1 ... A_k)$$

and in SC_2 we add the effect axiom

$$\Psi_2 = \forall s, a\ Holds(f, res(a, s)) \Rightarrow (Holds(\psi_{l+1}, s) \wedge a = A_{l+1}) \vee ... \vee (Holds(\psi_k, s) \wedge a = A_k) \vee (a = A_1 ... A_l) \vee (Holds(f, s) \wedge a \neq A_1 ... A_k).$$

First, notice that $\Psi \Rightarrow \Psi_1 \wedge \Psi_2$ because each of Ψ_1, Ψ_2 is a weakened version of Ψ (we weakened the consequent of Ψ to get each of them). Then, notice that $\Psi_1 \wedge \Psi_2 \Rightarrow \Psi$. This is because we include a UNA for actions, which implies that $(a = A_{l+1} \vee ... \vee a = A_k) \Rightarrow \neg(a = A_1 \vee ... \vee a = A_l)$, and using the resolution rule (see (Genesereth and Nilsson, 1987)) we get Ψ. Thus,

$$\Psi_1 \wedge \Psi_2 \equiv \Psi.$$

For all SC^i that is a copy of SC_1 we add $SC^i.\Psi_1$. Also, for all SC^i that is a copy of SC_2 we add $SC^i.\Psi_2$.

Now, assume that SC' already includes all the explanation closure axioms as detailed above and that SC has no explanation closure. Let $[\![SC]\!]$ be the set of models of SC.

Let $\mathcal{M}' \in [\![SC']\!]$ a model of SC' together with the explanation closure axioms. Let $\mathcal{M} \in [\![SC]\!]$ be a matching model, as found by our previous section of the proof (*Only Effect Axioms*).

(Lin and Reiter, 1994) provided a model-minimization policy that gives semantics[3] to explanation closure axioms. Roughly, it says that $\mathcal{M}$ is minimal if there is no model $\mathcal{M}'$ which agrees with $\mathcal{M}$ on all actions a, fluents f and situations s for which $\mathcal{M} \models Holds(f,s) \equiv Holds(f, res(a,s))$ but that has fluent f, situation s and action a for which a is possible ($Poss(a,s)$) in $\mathcal{M}$ and $\mathcal{M} \models Holds(f,s) \equiv \neg Holds(f, res(a,s))$ but for which $\mathcal{M}' \models Holds(f,s) \equiv Holds(f, res(a,s))$. If there is such a model, $\mathcal{M}'$, we write $\mathcal{M}' \prec \mathcal{M}$. This minimization is for models that satisfy the effect axioms, the UNAs for actions and the foundational axioms for situation calculus (inclusion of other formulae is done only after the minimization is complete; the process, called *filtering*, was first discussed by Sandewall (Sandewall, 1989; Sandewall, 1994)).

We use this minimization policy to show that our $\mathcal{M}'$ has a corresponding model in the sense of our first case above (*Only Effect Axioms*). If $\mathcal{M}$ is not minimal, let $\widetilde{\mathcal{M}}$ such that $\widetilde{\mathcal{M}} \prec \mathcal{M}$. Assume that $\widetilde{\mathcal{M}}$ differs from $\mathcal{M}$ in $Holds(f, res(a_{i+1}, S_i))$ for a minimal $i \leq n$ (otherwise, $\widetilde{\mathcal{M}}$ still matches our $\mathcal{M}'$). Then $\widetilde{\mathcal{M}} \models Holds(f, S_i) \equiv Holds(f, res(a_{i+1}, S_i))$ while $\mathcal{M} \models Holds(f, S_i) \equiv \neg Holds(f, res(a_{i+1}, S_i))$.

If $f \in L(SC^{i+1})$, then $\mathcal{M}' \models SC^{i+1}.Holds(f, SC^{i+1}.S_i) \equiv \neg SC^{i+1}.Holds(f, SC^{i+1}.res(SC^{i+1}.a_{i+1}, SC^{i+1}.S_i))$ because of the similar property that holds in $\mathcal{M}$ and the fact that $\mathcal{M}', \mathcal{M}$ agree on f in S_i, S_{i+1}. However, recall that $\mathcal{M}'$ satisfies all the explanation closure axioms that are stated in SC'. Since $\widetilde{\mathcal{M}} \models SC$ and is $\prec$-smaller than $\mathcal{M}$ we get another frame axiom (for the exact state of $S_i^{\mathcal{M}}$) that we can compile into an explanation closure axiom for f. This axiom then implies that

$$\forall a, s\ Holds(f,s) \wedge \neg Holds(f, res(a,s)) \Rightarrow (\neg Holds(state\_in\_S_i, s) \vee a \neq a_{i+1}).$$

Recall our observation about the way we can split an explanation closure axiom into the two partitions in a seemless way. By this observation we should have added the proper explanation closure axioms to the copies of SC_1, SC_2 in SC'. This would have prevented us from having $\mathcal{M}'$ as a model of SC'. Contradiction.

If $f \notin L(SC^{i+1})$, then $\mathcal{M}' \models f_i \equiv f_{i+1}$. This guarantees that if $\mathcal{M}'$ and $\widetilde{\mathcal{M}}$ agree on the value of f in situation S_i then they also agree in situation S_{i+1} (where $\mathcal{M}, \mathcal{M}'$ did not agree). Let $\widetilde{\mathcal{M}}$ be such that $\widetilde{\mathcal{M}} \prec \mathcal{M}$ and the first S_i in which $\widetilde{\mathcal{M}}, \mathcal{M}$ disagree has the smallest index i among all models that are $\prec$-smaller than $\mathcal{M}$. We show that, for this i, $\mathcal{M}'$ and $\widetilde{\mathcal{M}}$ agree on the value of f in situation S_i. This will show by induction that there is no situation S_j in which $\mathcal{M}', \widetilde{\mathcal{M}}$ disagree (on the fluent-situation combinations that count, namely, those mentioned in the previous section of our proof).

Otherwise, $\mathcal{M}', \widetilde{\mathcal{M}}$ disagree on f in S_i and thus $\mathcal{M}', \mathcal{M}$ agree on f in S_i. Take $j \leq i$ such that j is the smallest index for which $\mathcal{M}', \mathcal{M}$ disagree on f in S_j and $SC^j, ..., SC^i$ are copies of the same partition of SC. Similarly, take $k \geq i$ such that k is the largest index for which $\mathcal{M}', \mathcal{M}$ disagree on f in S_k and $SC^i, ..., SC^k$ are copies of the same partition of SC. If there is no such j, then $\widetilde{\mathcal{M}} \not\prec \mathcal{M}$ because $\mathcal{M}' \models f_i \iff f_{i+1}$ and similarly does $\mathcal{M}$ (because we did not find such j) while $\widetilde{\mathcal{M}}$ does have a change between $Holds(f, S_i)$ and $Holds(f, S_{i+1})$. Thus, there are such j and k.

Define $\widehat{\mathcal{M}}$ to be identical to $\mathcal{M}$ but with the $Holds(f, S_l)^{\widehat{\mathcal{M}}} \equiv f_l^{\mathcal{M}'}$ for all l such that $j \leq l \leq k$. $\widehat{\mathcal{M}} \prec \mathcal{M}$ because in $\widehat{\mathcal{M}}$ there is no change in the value of f throughout the situations $S_j, ..., S_k$ whereas there is at least one such change for S_i in $\mathcal{M}$. Furthermore, $\widehat{\mathcal{M}} \models SC$ because it satisfies all the effect axioms for all unchanged situations and fluents and there is no effect axiom that constrain f in those situations (the actions we take in $S_j, ..., S_k$ are in the other partition of SC, so f does not show in their effect axioms (recall that in this part of the proof SC includes *only effects axioms*)).

This means that we found a model of SC that contradicts our choice of $\widetilde{\mathcal{M}}$ (we chose it to have the first index i such that $\widetilde{\mathcal{M}}, \mathcal{M}$ disagree on S_i). Thus, if $\mathcal{M}$ is not $\prec$-minimal then we can find a $\prec$-smaller model that

[3] Other semantics are sometimes used for nondeterministic actions (e.g., (Levesque et al., 1997)). The following proof proceeds similarly for those.

agrees with $\mathcal{M}'$. This, together with an assumption of well-foundedness of $\prec$ (also called smoothness (Kraus et al., 1990)), which is assumed by (Lin and Reiter, 1994), provides our result: For every model $\mathcal{M}'$ of SC' there is a corresponding $\prec$-minimal model of SC. As a result, if $\widetilde{SC}$ is the set of effect axioms (SC) together with the explanation closure axioms, then for every $\mathcal{M}' \models SC'$ there is $\mathcal{M} \models \widetilde{SC}$ such that if $\mathcal{M}' \models SC^i.Holds(\psi, SC^i.res(SC^i.a_i, ..., SC^i.res(SC^i.a_1, SC^i.S0)...))$, for $\psi \in \mathcal{L}(SC^i)$, then $\mathcal{M} \models Holds(\psi, S_i)$.

Adding Observations The overall process that leads to the solution to the frame problem along the lines of successor-state axioms first computes those successor state axioms from effect axioms and state constraints (in the form of ramification constraints) and only then includes observations.

As a result, the pure deductive approach that we are taking here cannot treat state constraints. Any state constraints that we add are treated as observations.

For the case of observations (no uncompiled state constraints), assume that we have our set of explanation closure axioms already compiled into SC and SC'. Adding the observations at the proper places is analogous to removing models from each one of them (those that contradict the observations). These observations do not need to be deterministic, but each sentence should be expressible in the language of a single situation and a single partition from SC. Then, we get the previous result about $\mathcal{M}'$ having a corresponding $\mathcal{M}$ immediately. ■

References

Amir, E. (2000). (De)Composition of situation calculus theories. In *Proc. National Conference on Artificial Intelligence (AAAI '00)*, pages 456–463. AAAI Press/MIT Press.

Amir, E. (2001). Efficient approximation for triangulation of minimum treewidth. In *Proc. Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI '01)*, pages 7–15. Morgan Kaufmann.

Amir, E. (2002a). Projection, filtering and belief state representation in nondeterministic dynamic systems. Submitted for publication. Available on the author's website.

Amir, E. (2002b). Projection in decomposed situation calculus (with proofs). Available on the author's website.

Amir, E. and Maynard-Reid II, P. (1999). Logic-based subsumption architecture. In *Proc. Sixteenth International Joint Conference on Artificial Intelligence (IJCAI '99)*, pages 147–152.

Amir, E. and McIlraith, S. (2000). Paritition-based logical reasoning. In *Principles of Knowledge Representation and Reasoning: Proc. Seventh Int'l Conference (KR '2000)*, pages 389–400. Morgan Kaufmann.

Baral, C., Kreinovich, V., and Trejo, R. (2000). Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence*, 122(1-2):241–267.

Boutilier, C., Reiter, R., and Price, B. (2001). Symbolic dynamic programming for first-order MDPs. In *Proc. Seventeenth International Joint Conference on Artificial Intelligence (IJCAI '01)*, pages 690–697. Morgan Kaufmann.

Bylander, T. (1994). The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1–2):165–204.

Darwiche, A. (1998). Model-based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research*, 8:165–222.

Dechter, R. and Pearl, J. (1989). Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366.

Finzi, A., Pirri, F., and Reiter, R. (2000). Open world planning in the situation calculus. In *Proc. National Conference on Artificial Intelligence (AAAI '00)*, pages 754–760. AAAI Press.

Frank, J., Johnsson, A. K., and Morris, P. (2000). On reforumlating planning as dynamic constraint satisfaction. In Choueiry, B. and Walsh, T., editors, *Abstraction, reformulation and approximation, Proceedings of the 4th international symposium (SARA'2000)*, volume 1864 of *LNAI*, pages 271–280. Springer-Verlag.

Genesereth, M. R. and Nilsson, N. J. (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers Inc.

Green, C. (1969). Theorem-proving by resolution as a basis for question-answering systems. In Meltzer, B., Michie, D., and Swann, M., editors, *Machine Intelligence 4*, pages 183–205. Edinburgh University Press, Edinburgh, Scotland.

Haas, A. R. (1987). The case for domain-specific frame axioms. In *proceedings of the 1987 workshop on the frame problem*, pages 343–348.

Kraus, S., Lehmann, D., and Magidor, M. (1990). Nonmonotonic Reasoning, Preferential Models and Cumulative Logics. *Artificial Intelligence*, 44(1):167–207.

Lansky, A. (1988). Localized event-based reasoning for multiagent domains. *Computational Intelligence Journal, Special Issue on Planning*, 4(4):319–340.

Lansky, A. L. and Getoor, L. C. (1995). Scope and abstraction: Two criteria for localized planning. In *Proc. Fourteenth International Joint Conference on Artificial Intelligence (IJCAI '95)*, pages 1612–1618, Montreal, Canada.

Levesque, H., Reiter, R., Lesprance, Y., Lin, F., and Scherl, R. (1997). Golog: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31:59–84.

Lin, F. (1996). Embracing causality in specifying the indeterminate effects of actions. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*.

Lin, F. and Reiter, R. (1994). State constraints revisited. *Journal of Logic and Computation, Special Issue on Actions and Processes*.

Lin, F. and Reiter, R. (1995). How to progress a database II: The STRIPS connection. In *Proc. Fourteenth International Joint Conference on Artificial Intelligence (IJCAI '95)*, pages 2001–2007, Montreal, Canada.

McCarthy, J. and Hayes, P. J. (1969). Some Philosophical Problems from the Standpoint of Artificial Intelligence. In Meltzer, B. and Michie, D., editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press.

McIlraith, S. (1997). Representing action and state constraints in model-based diagnosis. In Senator, T. and Buchanan, B., editors, *Proc. National Conference on Artificial Intelligence (AAAI '97)*, pages 43–49, Menlo Park, California. American Association for Artificial Intelligence, AAAI Press.

McIlraith, S. (2000). Integrating actions and state constraints: a closed-form solution to the ramification problem (sometimes). *Artificial Intelligence*, 16(1–2):87–121.

McIlraith, S. and Amir, E. (2001). Theorem proving with structured theories. In *Proc. Seventeenth International Joint Conference on Artificial Intelligence (IJCAI '01)*, pages 624–631. Morgan Kaufmann.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann.

Pednault, E. P. D. (1989). ADL: exploring the middle ground between STRIPS and the situation calculus. In *Proc. First International Conference on Principles of Knowledge Representation and Reasoning (KR '89)*, pages 324–332.

Pfeffer, A. (2001). Sufficiency, separability and temporal probabilistic models. In *Proc. Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI '01)*, pages 421–428. Morgan Kaufmann.

Pinto, J. (1999). Compiling ramification constraints into effect axioms. *Computational Intelligence*, 15(3):280–307.

Pinto, J. A. (1994). *Temporal Reasoning in the Situation Calculus*. Ph.d. dissertation, Department of Computer Science, University of Toronto, Toronto. Also available as Technical Report Number KRR-TR-94-1.

Reiter, R. (1991). The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Lifschitz, V., editor, *Artificial Intelligence and Mathematical Theory of Computation (Papers in Honor of John McCarthy)*, pages 359–380. Academic Press.

Reiter, R. (1992). The projection problem in the situation calculus. In *Proceedings AIPS*, pages 198–203.

Reiter, R. (2001). *Knowledge In Action: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press.

Sandewall, E. (1989). Filter preferential entailment for the logic of action in almost continuous worlds. In *IJCAI-89*.

Sandewall, E. (1994). *Features and Fluents*. Oxford University Press.

Santibaez, J. S. (1999). Declarative formalization of reasoning strategies: a case study on nonlinear planning. In Golumbic, M., Morgenstern, L., and Shimoni, E., editors, *Bar Ilan Symposium on the Foundations of Artificial Intelligence*. http://www-formal.stanford.edu/jsierra/aaai-draft-98-s.ps.

Schubert, L. K. (1990). Monotonic solution of the frame problem in the situation calculus. In Kryburg, H., Loui, R., and Carlson, G., editors, *knowledge representation and defeasible reasoning*, pages 23–67. Kluwer.

Shanahan, M. (1997). *Solving the Frame Problem, a mathematical investigation of the common sense law of inertia*. MIT press, Cambridge, MA.

Observations and the Probabilistic Situation Calculus

Paulo Mateus
CLC - Dep. Mathematics
Instituto Superior Técnico
Av. Rovisco Pais
1049-001 Lisboa

António Pacheco
CMA - Dep. Mathematics
Instituto Superior Técnico
Av. Rovisco Pais
1049-001 Lisboa

Javier Pinto *

Abstract

In this article we propose a Probabilistic Situation Calculus logical language to represent and reason with knowledge about dynamical worlds in which actions have uncertain effects. Two essential tasks are addressed when reasoning about change in worlds: Probabilistic Temporal Projection and Probabilistic Belief Update. Uncertain effects are modeled by dividing an action into two subparts: a deterministic input (agent produced) and a probabilistic reaction (nature produced). The probability distributions of the reactions are assumed to be known. Our logical language is an extension to Situation Calculae in the style proposed by Raymond Reiter. There are three aspects to this work. First, we extend the language to accommodate terms dealing with belief and probability. Second, we provide a operational semantics based on Randomly Timed Automata. Finally, we develop Monte-Carlo algorithms to efficiently interpret the probability and belief terms. With the framework proposed we discuss how to develop a reasoning system in Mathematica capable of performing temporal projection and belief update in the Probabilistic Situation Calculus. Finally, we present a sound basis to set rewards and observation planning.

BACKGROUND AND OBJECTIVES

Lately, there has been increasing interest in improving the expressive power of logical languages for representing knowledge about dynamic worlds with non-determinism and uncertainty (e.g., [Pinto et al 2000, Bacchus et al 1999, Poole 1998, Grosskreutz and Lakemeyer 2000, Baral and Tuan 2001]). In this article, we extend previous work on the *Probabilistic Situation Calculus* [Pinto et al 2000, Mateus et al 2001], a logical language for knowledge representation and reasoning about dynamic worlds in which actions have uncertain effects. In this article, we:

- Show how to represent knowledge about actions that have uncertain effects by exploiting the separation between *agent initiated actions* and nature's *random reaction*, first introduced in [Pinto et al 1999].
- Introduce observations as agent actions with uncertain effects. Thus, as in [Bacchus et al 1999] observations are noisy; e.g., we may observe that the outside temperature is 25 degrees, but the reading may have an error with a Gaussian distribution.

As discussed in [Grosskreutz and Lakemeyer 2001], there at least two essential tasks to be addressed when reasoning about change in worlds in which there are uncertain actions and noisy observations: [a] Probabilistic Temporal Projection: Given an initial state, a sequence of uncertain actions (an uncertain plan) and some fluent, determine the probability that the fluent will hold after the actions are executed in the initial state. [b] Probabilistic Belief Update: Given an initial state, a sequence of uncertain actions (an uncertain plan), an observation, and some fluent, determine the probability that the fluent will hold after the uncer-

* Unfortunately Javier Pinto passed away in an accident while this paper was being prepared. Formerly, he was at Bell Labs, Database Systems Research Dept., 600 Mountain Ave., New Jersey 07974, U.S.A.

tain actions are executed in the initial state, and an uncertain (or *noisy*) observation is performed.

An important characteristic of the language of the probabilistic situation calculus is the hybrid nature of its semantics. In particular the logic includes the reals, and its usual operators, without axiomatizing them. Thus, we assume, at the semantic level, a fixed interpretation for them, and the satisfaction relation has to take this into account. When reasoning with such a theory, logical reasoning has to be combined with reasoning about the reals. The latter is done by appealing to an oracle (in our case, we implement this oracle using MATHEMATICA). Furthermore, probability theory is assumed at the semantic level, and is not axiomatized. Our approach is radically different from approaches based on possible world semantics (e.g., [Bacchus et al 1999]), as discussed in the full article.

STATE TRANSITIONS

Our work extends the Situation Calculus in the style presented in [Reiter 1991, Reiter 2000], with the notion of uncertain actions. Recall that the situation calculus is a sorted logic with sorts $\mathcal{S}$ (situations), $\mathcal{A}$ (actions), $\mathcal{F}$ (fluents), etc. An essential component of any situation calculus theory is a description of how the world is affected by actions. This is done by means of effect axioms, which given a situation s, specify how the world would be in a situation $do(a, s)$. Here, a is an action term, while s and $do(a, s)$ are situation terms. The latter term denotes the situation that results from performing action a in situation s. For instance, if the action is $jump(x)$, meaning jump forward x meters, then one effect axiom could be:

$$holds(at(y - x), s) \supset holds(at(y), do(jump(x), s)). \quad (1)$$

At the heart of our proposal to integrating actions with uncertain outcomes, is the introduction of the sorts $\mathcal{I}$ (inputs) and $\mathbb{R}$ (reals), along with the requirement that any action corresponds to a pair $\langle i, r\rangle$ (an input and a real). The second component is referred to as a random *reaction*. For instance, we can introduce uncertainty in the previous example by having $jump_i(x)$ as an input, and the reaction being the actual distance jumped. Thus, the pair $\langle jump_i(3), 3.4\rangle$ would denote the action of jumping forward 3 meters but actually jumping 3.4 meters. The effect axiom (1) would now look like:

$$holds(at(y - r), s) \supset holds(at(y), do(\langle jump_i(x), r\rangle, s)). \quad (2)$$

As a second example, we model a dice throw as six different actions: one input (*throw*) and six possible reactions (the reals $1, 2, 3, 4, 5$ and 6). Pictorially: Thus,

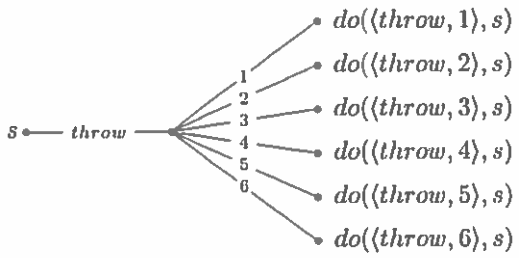


Figure 1: Throwing a dice, and possible reactions

an agent's input (e.g., *throw*), has reactions that are indeterminate. Probability distributions are placed over the reactions. In the dice example, the distribution would be discrete, and in the *jump* example the distribution would be continuous. However, in order to have a uniform theory, without having to explicitly distinguish between continuous and discrete cases, we approximate discrete distributions by continuous ones that are described using the *Dirac Delta* function. Recall that the Dirac Delta of a real x is a function that is zero over $\mathbb{R}$ except in an infinitesimal neighborhood around x, whose indefinite integral is 1. Thus, the probability density function in the case of the dice example, would be

$$dice(x) = \sum_{i=1}^{6} 1/6 \text{ DiracDelta}(i)(x).$$

This approach is completely general and allows for mixed distributions; i.e., inputs with reactions that have distributions that are continuous in some areas and discrete in others. Given that we have divided actions into its *input* and *reaction* components, we have decomposed the *Poss* relation[1] into *iposs* and *rposs*. Thus, we write:

$$Poss(\langle i, r\rangle, s) \equiv iposs(i, s) \wedge rposs(r, i, s). \quad (3)$$

iposs is similar to the standard *Poss* predicate. $rposs(r, i, s)$ is true when reaction r is possible after input i is performed in situation s. In the dice example,

$$rposs(r, throw, s) \equiv dice(r) > 0.$$

As discussed in the sequel, we actually use the *cummulative distribution function (cdf)* instead of the density

[1] In the situation calculus, $Poss(a, s)$ is a relation that holds when action a can be executed in situation s.

function. In general, we write $cdf(i, s)$ to denote the cdf of the reaction to input i in situation s. Thus $cdf(throw, s)(x)$ has value zero for values of x smaller than 1, value 1 for x greater or equal than 6, value $1/6$ when $1 \leq x < 2$, value $2/6$ when $2 \leq x < 3$, etc.

Notice that we are assuming the structure of the reals ($\mathbb{R}$) along with a very rich arsenal of real operators. We do not attempt to axiomatize $\mathbb{R}$, instead we assume standard interpretations for $\mathbb{R}$ and for the real operators. Furthermore, for reasoning with the resulting theories, we assume the existence of an *oracle* for computations involving the reals. In particular, as shown in examples later, we take MATHEMATICA to be an implementation of this oracle. In essence, our approach mixes *logical reasoning* with *analytical calculus*.

Getting back to the dice example, if one wanted to compute the probability that some fluent hold after input i is performed in a situation s, we would have to compute:

$$prob(f, i, s) = \int_{-\infty}^{+\infty} cf(holds(f, do(\langle i, x\rangle, s)))cdf(i, s)(dx), \quad (4)$$

where the integral is of Lebesgue-Stieltjes type [Resnick 1999] and cf stands for characteristic function (1 if its argument is true, and zero otherwise).

TREE STRUCTURE

In the previous section, we concentrated on describing the approach to model uncertain actions, which derives from [Pinto et al 2000] (where only discrete distributions were considered), and extensions, presented in [Mateus et al 2001], to deal with continuous distributions. We have basically addressed the problem of describing the uncertainty that arises after inputs are performed in a known situation. Notice that a single input, in a fixed situation, gives rise to a potentially infinite set of possible successor situations. For instance, in formula (2), the distribution for the reaction r to a $jump(x)$ input can be a normal[2] $\mathtt{N}(0, 1)$; in which case the possible situations after a $jump(x)$ input would be infinite.

Let us consider now what happens when several inputs are performed in sequence[3]. For instance, when a dice is thrown several times in succession. Naturally we would be interested in computing the probability that some fluent holds after such a sequence is performed. In order to be able to pose such a query, we introduce input sequences as elements of discourse via a new sort $\mathcal{I}^*$ (a generic variable of type $\mathcal{I}^*$ will be $\vec{i}$). For instance, suppose that we define the fluent *Close* as being at a distance of less than 1 from the origin:

$$holds(Close, s) \equiv holds(at(y), s) \wedge -1 < y < 1,$$

Then, we can ask what is the probability that *Close* is true after some sequence of inputs (e.g., $\{jump_i(0.3), jump_i(0.3), jump_i(0.3)\}$) is performed.

A sequence of inputs and a situation define a *Markov Process*. Thus, we assume that the distribution of the outcomes resulting from an input depends only on the situation on which the input is performed. A situation s and an input sequence $\vec{i}$ define a subtree of the original situation structure. We can say that a situation s' is *mp-accessible* from a situation s and an input sequence $\vec{i}$, when s' is reached at the end of some evolution of the Markov process induced by performing the sequence of inputs $\vec{i}$ in the situation s. For instance, the situation $do(\langle jump_i(0.3), 0.25\rangle, do(\langle jump_i(0.3), 0.4\rangle, S_0))$ is *mp-accessible* from situation S_0, with the input sequence $\{jump_i(0.3), jump_i(0.3)\}$. The Markovian assumption lets us generalize (4). Thus we introduce an inductive definition of *prob* on the input sequence $\vec{i}$.:

$$\begin{aligned} &\vec{i} = \epsilon \supset prob(f, \vec{i}, s) = cf(holds(f, s)), \\ &(\exists i_0, \vec{i}_+).\vec{i} = (i_0; \vec{i}_+) \supset \\ &\quad prob(f, \vec{i}, s) = \\ &\qquad \int_{-\infty}^{+\infty} prob(f, \vec{i}_+, do(\langle i_0, x\rangle, s))cdf(i_0, s)(dx). \end{aligned} \quad (5)$$

Here x represents all the outcomes that can arise from performing i_0 in situation s. For each such outcome, we then consider all the outcomes that result from performing the remaining part of the input sequence.

THE INITIAL SITUATION(S)

In the same spirit as the situation calculus, the probabilistic situation calculus allows one to write theories to specify hypothetical developments from the initial situation S_0. Temporal projection provides answers to questions of the form: *Does fluent f hold after a specific sequence of actions is performed in* S_0. On the other hand, *probabilistic* temporal projection, in the framework of the probabilistic situation calculus, seeks to answer questions of the following forms:

[2] We use $\mathtt{N}(\mu, \sigma)$ to denote the *cdf* of the Normal distribution with the usual parameters. Also, later, we use $U(\cdot, \cdot)$ to denote the cdf of the uniform distribution.

[3] In previous work, this problem was only addressed in the semantical and computational aspects

- *What is the probability that some fluent f holds after a sequence of* inputs *is performed starting in S_0?*. This question is only meaningful if this probability is discrete.
- *What is the probability that a fluent $f(x)$, holds after a sequence of* inputs *is performed starting in S_0, and for some value of x in the range x_0 to x_1?*. We have implicitly assumed that the argument x is of type $\mathbb{R}$.

In the context of probabilistic temporal projection, we should consider cases in which an axiomatizer might want to specify a set of situations as probable starting situations. In contrast, a feature of the situation calculus is that the world is assumed to have a single initial situation, called S_0. This means that the agent starts in a completely defined world with no uncertainty about what is true or not. In general, however, agents may not have complete knowledge about the starting state. Instead, an agent may have a family of situations as possible starting situations. There are two natural ways in which one can avoid this initial determinism of the situation calculus. First, one can consider an explicit family of initial situations (as in [Bacchus et al 1999]) and, if one were so inclined, a probability distribution could be defined over these situations. Second, one could define a *phantom* starting situation, and a starting input that generates all the situations that are possible. Hence, the initial situation S_0 would not represent any actual situation. Rather, it represents an initial time in which a phantom *big bang* generates the situations that can be regarded as representing the initial state of the world (as modeled by the theory).

Recall that in the situation calculus a special relation $Poss \subseteq Actions \times Situations$ is used to estabish conditions under which actions can or cannot be executed. Since now we have split actions into subparts, we also separated $Poss$. We use $iPoss \subseteq Input \times Situations$ to express the conditions necessary for inputs to be possible, and $rposs \subseteq Input \times Reals \times Situations$ to express the conditions under which specific reactions are possible after an input is performed in a situation. Thus, to model the big bang, we will need:

$$iPoss(i, s) \supset [i = BigBang \equiv s = S0]$$

Now, as an example, imagine defining some object's position with respect to the X axis. We can have the following:

$$Poss(\langle BigBang, x\rangle, s) \supset at(x, do(\langle BigBang, x\rangle, s)),$$
$$cdf(BigBang, S_0) = \mathtt{N}(\mu_x, \sigma_x).$$

If we wanted to introduce a more complex initial scenario, with more properties and different distributions, the *BigBang* would have to be an input with multivariate reactions (as in a later example).

OBSERVATIONS AS CONDITIONS

In this section we define formally the *probabilistic temporal projection with observations*, which one might also call *Belief Updates*. The problem can be seen as solving probabilistic temporal projection (as informally defined in the previous section) but conditioned on one or more observations along the input sequence. To ease the presentation and for lack of space, we only consider a simplified case in which there is a single observation performed as a last input in the input sequence.

Observations are introduced into the logical language as a subsort $\mathcal{O}$ of $\mathcal{I}$. Thus, $\mathcal{O} \subseteq \mathcal{I}$, for convenience, we also introduce a sort predicate $observ = \mathcal{O}$. Notice that observations, together with their outcomes, are regular actions (inputs followed by reactions). For technical reasons (that we explain later), we introduce a fluent $Obs : \mathbb{R} \rightarrow \mathcal{F}$, such that if o is an observation (i.e., $o \in \mathcal{O}$):

$$holds(Obs(r), do(\langle o, r\rangle, s))$$

That is, *Obs* registers the outcome of the last observation performed. Furthermore, if $i \in \mathcal{I}\backslash\mathcal{O}$, then the only effect of observations is described with:

$$\neg observ(i) \supset holds(Obs(0), do(\langle i, r\rangle, s)).$$

Thus, after an input that is not an observation, the observed value is an arbitrary constant (e.g., 0). Notice that these axioms are domain independent.

Assuming that changes in the world are described using effect axioms, and the frame problem solved by using explanation closure axioms, then the resulting theory will lead to the conclusion that observations don't affect any fluents other than *Obs*. Indeed, the result of the completion should be that:

$$f \neq Obs(x) \supset holds(f, do(\langle o, x\rangle, s)) \equiv holds(f, s)$$

The axiom states that the state of the world does not change as result of performing observations. Note that all variables are implicitly universally quantified.

Example 1 Let us consider the simplest possible example with observations. Assume that there is a coin inside a box, and an unreliable witness tells us whether it is heads or tails. We believe this witness with probability 0.7. We might want to know what is the probability that the coin is actually tails when the witness

says so. To set things up, we have a theory with a single fluent *Tails* which holds only if *Tails* is up[4].

$$Poss(\langle BigBang, r\rangle, s) \supset [holds(Tails, s) \equiv r = 1]$$
$$cdf(BigBang, s)(x) = Bernoulli(0.5)(x),$$
$$cdf(ObsTails, s)(x) = Bernoulli(0.7cf(holds(Tails, s)) + 0.3(1\text{-}cf(holds(Tails, s))))(x).$$

By using (5) one can verify that the probability that *Tails* holds after the unitary input sequence $\{BigBang\}$ is the same as the probability that *Tails* holds after the input sequence $\{BigBang, ObsTails\}$. However, as we see later, conditioning on the observation (i.e., computing the probabiity that *Tails* holds after $\{BigBang, ObsTails\}$ and seeing the outcome of $ObsTails$), allows us to do *Bayesian belief updates.*

Notice that observations are handled in a radically different manner than in proposals based on epistemic logics (e.g., [Bacchus et al 1999]). Basically, an observation action simply changes the nature of the distribution over the space of situations reached after performing actions. Thus, the distribution over the space of situations possible after input sequence $\vec{i}$ is changed after an observation o is made.

As in the previous example, the rest of the article considers only univariate outcomes for all the inputs. However, it is not difficult to extend this approach to multivariate cases (extending the presentation in [Mateus et al 2001] with observations).

Now, we extend the vocabulary of the probabilistic situation calculus to deal with updates based on observations (we deal with semantic issues in a later section). We introduce the operator $bel : \mathcal{F} \times \mathcal{I}^* \times \mathcal{S} \times \mathcal{O} \times \mathbb{R} \rightarrow \mathcal{P}$, with the following syntactic sugar: The term $bel(f, \vec{i}, s|o = r)$, denotes the probability that f holds, after input $\vec{i}$ is performed starting in s, and observing (at the end) reaction r to random observation o. We need to add an axiom to define[5]:

$$bel(f, \vec{i}, s|o = r) = \frac{prob(f \wedge Obs(r), (\vec{i}; o), s)}{prob(Obs(r), (\vec{i}; o), s)} \quad (6)$$

[4]Here, Bernoulli represents the following cumulative distribution function:

$$Bernoulli(p)(y) = \begin{cases} 0 & \text{whenever } y < -1 \\ 1-p & \text{whenever } -1 \leq y < 1 \\ 1 & \text{otherwise} \end{cases}$$

[5]This axiom is based on the definition of conditional probability $P(A|B) = P(A \wedge B)/P(B)$.

In this expression we have used $f \wedge Obs(r)$ as syntactic sugar for $and(f, Obs(r))$; where $and : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{F}$ is a fluent operator. Here, if f_1, f_2 are fluent, $and(f_1, f_2)$ is a *defined* fluent (see [Pinto 1994]).

Note that to evaluate *bel* in (6), we need to evaluate two *prob* terms using (5). The expression (5) is a recursive definition for *prob*. By simple inspection, it turns out that to evaluate an expression like $prob(f, \vec{i}, s)$, where the input sequence $\vec{i}$ has dimension n requires solving an n-dimensional indefinite integral. Except for trivial cases, this integration is impossible to solve in exact terms. Before we discuss how we address this difficulty, we will discuss the semantics underlying the probabilistic situation calculus with observations.

EXTENDED PROBABILISTIC SITUATION CALCULUS SEMANTICS

In this section, we introduce the semantics for the probabilistic extension to the situation calculus. We specify how the situation structure is generated and how the probability distributions are laid on top of this structure. The presentation of the semantics for the probabilistic situation calculus provided here is an extension to the Randomly Reactive Automata presented in [Mateus et al 2001]. In this extended abstract, we do not present the full semantics and leave standard aspects of Tarskian semantics out.

We provide a probabilistically sound specification for the computation of beliefs and probabilities based on the Randomly Reactive Automata. This should be understood as an *operational semantics*, in the sense that it specifies a mechanism to evaluate the prior and posterior probabilities (of fluents being true before or after certain observations). Notice that any system that reasons about these probabilities (like our oracle MATHEMATICA) has to obey this semantic specification.

As described formally below, in a Randomly Reactive Automaton with observations, actions are represented as pairs from $\mathcal{I}\mathbb{R}$, where $\mathcal{I}$ is the set of inputs (part of the specification of the Randomly Reactive Automaton), and $\mathbb{R}$ are the real numbers. These pairs correspond exactly to the input-reaction pairs of the last section. The situations are not explicitly introduced because they are simply sequences of $\mathcal{I}\mathbb{R}$ elements. Thus, we can correctly define the universe of situations as $S = (\mathcal{I}\mathbb{R})^*$. The empty string corresponds to the initial situation. We formally define *Randomly*

Reactive Automata with observations as a tuple

$$\mathcal{A} = \langle I, O, F, T, M, g, h \rangle$$

where:

- I is a set of *inputs*. $O \subseteq I$ is a set of *observations*. F is a set of (proper) *fluents* (as specified in the full semantics, this set excludes the special fluent *Obs*).
- $T = \{T^f\}_{f \in F}$ where $T^f \subseteq S$ is such that

  $$T^f_{\vec{i}\,s} = \{\vec{x} \in \mathbb{R}^n : s i_1 x_1 \ldots i_n x_n \in T^f\}$$

  is Borel-measurable for all $\vec{i} = \langle i_1 \ldots i_n \rangle \in I^*$ and $s \in S$. For each fluent $f \in F$, T^f represents the set of situations in which f holds.
- $M = \{M^{[s]}\}_{[s] \in [S]}$ where $M^{[s]} \subseteq I$ is the set of *possible inputs* at $[s]$, and $[S] = S/\approx$, $[s] = \{s' \in S : s \approx s'\}$ and $\approx$ is an equivalence relation of *indistinguishability* over S such that $s_1 \approx s_2$ iff $s_1 \in T^f \equiv s_2 \in T^f$ for all $f \in F$.
- $g = \{g^{[s]}_i\}_{i \in I \setminus O, [s] \in [S]}$, where $g^{[s]}_i$ is a generalized density function [Mateus et al 2001] of the *(hidden) reaction after performing input i at $[s]$*;
- $h = \{h^{[s]}_o\}_{o \in O, [s] \in [S]}$, where $h^{[s]}_o$ is a generalized density function of the *(observed) reaction after performing observation o at $[s]$*;

In order to completely specify the structures for probabilistic situation calculus interpretations, we need to add sets of objects for all domain sorts, along with sets of symbols for domain predicates and functions. Also, we need to specify the interpretation for the *prob* and *bel* predicates, which are not fully axiomatized and play a special role in the theory. This latter part is completed in the next section. Armed with these interpretation structures it is relatively straightforward to provide a Tarskian style specification for the satisfaction relation between a Probabilistic Situation Calculus interpretation structure and a Probabilistic Situation Calculus formula. Moreover, as expected, first order reasoning is sound with respect to the proposed semantics. Both the Tarskian specification for the satisfaction and the soundness results are deferred to the full paper.

BELIEF

In this section we introduce the interpretation for the *prob* and *bel* predicates in the light of randomly timed automata. In general, such interpretation requires solving an arbitrary n-dimensional indefinite integral, which is unfeasible via classical analytic calculus. To overcome this hurdle, we present a Monte-Carlo based method for approximating such integrals and discuss its efficiency for the applications in mind.

Note that, if no observation is made, the *belief* that a fluent f holds after executing a sequence of actions $\vec{i}$ at a situation s is just given by the probability $prob(f, \vec{i}, s)$ as defined in (5). Given a Randomly Reactive Automata $\mathbf{A}$ we fix the value of $prob(f, \vec{i}, s)$ by calculating the probability of reaching a situation in T^f by a sequence of inputs denoted by $\vec{i} = i_1 \ldots i_n$ from the situation denoted by s, more precisely:

$$[\![prob(f, \vec{i}, s)]\!]_{\mathbf{A}} = \int_{\mathbb{R}^n} 1_{T^f_{\vec{i}\,s}}(\vec{x}) \prod_{k=0}^{n-1} g^{[s\vec{i}_k|\vec{x}_k|]}_{i_{k+1}}(x_{k+1}) dx_n \ldots dx_1 \quad (7)$$

where $1_{T^f_{\vec{i}\,s}}$ is the characteristic function of $T^f_{\vec{i}\,s}$ and $s\vec{i}_k|\vec{x}_k] = s_1 i_1 x_1 \ldots i_k x_k$. This interpretation is sound with respect to the axiomatization presented in (5).

Unfortunately, solving this kind of integrals with all generality is an herculean task, and therefore methods of approximation are required. Literature, such as [Evans and Swartz 2000], clearly indicates that Monte-Carlo based methods are the most suitable for dealing with such probabilities. The overall idea is to sample over m independent random situations reached by $\vec{i}$ from s and take the rate of those that reach T^f to approximate the probability. In practice, this sampling can be easily obtained via a pseudo-random generator, which makes easy to encode this function in any language with this utility, like for instance, MATHEMATICA.

Proposition 1 The value $[\![\text{prob}(f, \vec{i}, s)]\!]_{\mathbf{A}}$ can be estimated via a Monte-Carlo approach by

$$\widehat{\text{prob}}_m(f, \vec{i}, s) = \frac{\sum_{k=1}^{m} 1_{T^f_{\vec{i}\,s}}(\vec{x}_k)}{m}$$

where $\{\vec{x}_k\}_{k=1,\ldots,m}$ are m vectores of reactions obtained by a Monte-Carlo independent sampling of reactions obtained after executing input $\vec{i}$ from s. Moreover $\widehat{\text{prob}}_m(f, \vec{i}, s)$ is a consistent estimator of $\text{prob}(f, \vec{i}, s)$; that is, it converges with probability 1 to $[\![\text{prob}(f, \vec{i}, s)]\!]_{\mathbf{A}}$.

An advantage of the proposed Monte-Carlo algorithm is that, fixed s and $\vec{i}$, it is polynomial in the size of

the sampling m. Therefore, if we want a more precise approximation, it is feasible just to consider more samples. An important addon result is that the (random) estimation error of this approximation has an asymptotic Gaussian distribution with mean 0 and standard deviation given by:

$$\sqrt{\frac{p(1-p)}{m}}$$

where $p = [\text{prob}(f, \vec{i}, s)]_{\mathbf{A}}$.

A more precise notion of whether a fluent f holds or not might be achieved if an observation o is made after executing $\vec{i}$ from s. Naturally, we only gain any information with the observation if the latter is not *independent* from the fluent holding. If the outcome of observing o is independent from fluent f holding the following bad thing happens

$$bel(f, \vec{i}, s|\ o = r_1) = prob(f, \vec{i}, s),$$

meaning that observing o is totally useless to determine f. With our framework it is possible to measure the information gained by performing an observation o in order to know if a fluent f holds or not. This problem is of utmost relevance when observations are expensive and a plan of observations has to be set up. The full treatment of this problem is deferred to an extended version of this paper.

Independently of the value that an observation might provide to determine whether a fluent is holding or not, it is always possible to define in rigorous terms the *belief* of a fluent holding given an observation. Indeed, as presented for the discrete case in (6), for interpreting $bel(f, \vec{i}, s|o = r_1)$ we have to compute the conditional probability of reaching T^f by $\vec{i}$ from s given that observing o resulted in the value r_1. Hence, for a Randomly Reactive Automata $\mathbf{A}$ we have $[bel(f, \vec{i}, s|\ o = r_1)]_{\mathbf{A}} = \lim_{y \to +\infty}$

$$\frac{\int_{(-y,y)^k} 1_{T^f_{\vec{i},s}}(\vec{x}) h_o^{[s\vec{i}_n]\vec{x}_n]}(r_1) \prod_{k=0}^{n-1} g_{i_{k+1}}^{[s\vec{i}_k]\vec{x}_k]}(x_{k+1}) dx_n..dx_1}{\int_{(-y,y)^k} h_o^{[s\vec{i}_n]\vec{x}_n]}(r_1) \prod_{k=0}^{n-1} g_{i_{k+1}}^{[s\vec{i}_k]\vec{x}_k]}(x_{k+1}) dx_n..dx_1}$$

Once again, solving the integrals presented above is an unfeasible task. However, contrasting with *prob*, approximation of conditional probabilities through Monte-Carlo methods is not straightforward, as we shall see.

Proposition 2 The value $[bel(f, \vec{i}, s|\ o = r_1)]_{\mathbf{A}}$ can be estimated via a Monte-Carlo method by

$$\widehat{\text{bel}}_m(f, \vec{i}, s|\ o = r_1) = \frac{\sum_{k=1}^m 1_{T^f_{\vec{i},s}}(\vec{x}_k) h_o^{[s\vec{i}\vec{x}_k]}(r_1)}{\sum_{k=1}^m h_o^{[s\vec{i}\vec{x}_k]}(r_1)}$$

where $\{\vec{x}_k\}_{k=1,\ldots,m}$ are the first m vectors of reactions obtained by a Monte-Carlo independent sampling verifying $h_o^{[s\vec{i}\vec{x}_k]}(r_1) > 0$. Moreover $\widehat{\text{bel}}_m(f, \vec{i}, s|\ o = r_1)$ is a consistent estimator of $bel(f, \vec{i}, s|\ o = r_1)$.

The need to guarantee m samples where there is positive probability density of obtaining r_1 from observing o is a shortcoming of Proposition 2. Observe that, if the reaction value r_1 has no probability (density) of being observed and it is, something very bad has happened. In particular, it is a clear evidence that one has misspecified the stochastic behavior of the agent. In this latter case it is impossible to obtain the required m samples, since observing the value r_1 is *inconsistent* with the specification, and therefore no *belief update* can be achieved. Putting aside such pathological behaviours, it still might be very hard to find situations where there is positive probability density of obtaining r_1 from observing o. In detail, for a starting situation s and sequence of inputs $\vec{i}$, the probability of reaching such situations is

$$p(\vec{i}, s, o, r_1) = \int_{\mathbf{R}^n} 1_{\{\vec{x}: h_o^{[s\vec{i}\vec{x}]}(r_1) > 0\}}(\vec{x}) \prod_{k=0}^{n-1} g_{i_{k+1}}^{[s\vec{i}_k]\vec{x}_k]}(x_{k+1}) dx_n \ldots dx_1.$$

Note, comparing with (7), that this value can be approximated via Monte-Carlo in the same way as *prob*. So, given s, $\vec{i}$, o and r_1, the next question arises:

How big must a sample be in order to obtain m vectors in $\{\vec{x} : h_o^{[s\vec{i}\vec{x}]}(r_1) > 0\}$?

It is intuitive that the expected value for the number of samples is $m/p(\vec{i}, s, o, r_1)$, further introspection leads to the following result.

Proposition 3 Let $M_m(\vec{i}, s, o, r_1)$ be the random variable defined as

$$M_m(\vec{i}, s, o, r_1) = \inf \left\{ k : \sum_{j=1}^k 1_{\{\vec{x}: h_o^{[s\vec{i}\vec{x}]}(r_1) > 0\}}(\vec{X}_j) \geq m \right\}$$

where $\vec{X}_j = (X_{1j}, X_{2j}, \ldots, X_{|\vec{i}|j})$ with $\vec{X}_1, \vec{X}_2, \ldots$ being independent sequences of reactions obtained

after executing input $\vec{i}$ from s, and $|\vec{i}|$ denoting the cardinality of $\vec{i}$. Then $M_m \sim$ Negative Binomial $(m, p(\vec{i}, s, o, r_1))$; that is:

$$P(M_m(\vec{i}, s, o, r_1) = k) = \binom{k-1}{m-1}(1 - p(\vec{i}, s, o, r_1))^{k-m} p(\vec{i}, s, o, r_1)^m$$

for $k = m, m+1, \ldots$.

Thus, the random variable $M_m(\vec{i}, s, o, r_1)$ has expected value $m/p(\vec{i}, s, o, r_1)$ and variance $m(1 - p(\vec{i}, s, o, r_1))/p(\vec{i}, s, o, r_1)^2$. For large m, $M_m(\vec{i}, s, o, r_1)$ can be approximated by a normal distribution via

$$\frac{M_m(\vec{i}, s, o, r_1) - \frac{m}{p(\vec{i}, s, o, r_1)}}{\sqrt{\frac{m(1-p(\vec{i}, s, o, r_1))}{p(\vec{i}, s, o, r_1)^2}}} \simeq \mathrm{N}(0, 1)$$

where $\simeq$ stands for *approximate in distribution.*

Rewards can be introduced by considering a real valued (measurable) function from the set of indistinguishable situations. The reward of executing a plan of actions is obtained by approximating (via Monte-Carlo) the integral of such function over the probability measure associated to the plan. The information given by an observation for checking whether a fluent holds or not (value relevant for observation planning) can be approximated by this method, since the entropy of a sequence of actions given an observation can be obtained as a reward.

THE FROGBOT EXAMPLE

This is a simple example in which a frogbot can hop to the left or to the right, following a random exponential law (along a single dimension, starting at position 0). The frogbot is also equipped with a (faulty) observation device that indicates the position of the robot with an uniform random error.

SPECIFICATION

To model this example we use the sort $\mathcal{R}$ for real numbers, and introduce two constants *MoveLeft* and *MoveRight* of type $\mathcal{I}$, the constant *Find* of type $\mathcal{O}$, the function $pos : \mathcal{R} \to \mathcal{F}$, and the constants *Center* and *Hawk* of type $\mathcal{F}$. The axiomatization follows:

Axioms about the initial situation

$$holds(pos(0), S_0). \tag{8}$$

$$holds(obs(0), S_0). \tag{9}$$

Hence, we assume that the frogbot starts at position 0 and starts by observing 0.

CDF axioms:

$$iposs(MoveLeft, s) \wedge s \geq S_0 \supset cdf(MoveLeft, s) = Exponential(3); \tag{10}$$

$$iposs(MoveRight, s) \wedge s \geq S_0 \supset cdf(MoveRight, s) = Exponential(2); \tag{11}$$

$$iposs(Find, s) \wedge s \geq S_0 \supset cdf(Find, s) = Uniform(pos(s)). \tag{12}$$

Here, we refer to the uniform distribution over the interval $[pos(s) - 0.5, pos(s) + 0.5]$.

Action precondition axioms

Action precondition axioms, which, in Reiter's style of axiomatization, are expressed in terms of conditions for *Poss*, are now expressed in terms of *iposs*. I.e., we specify constraints for the possibility of giving some input. Thus, for this example, we write:

$$iposs(MoveLeft, s) \equiv holds(pos(x), s) \wedge x \geq 0; \tag{13}$$

$$iposs(MoveRight, s) \equiv holds(pos(x), s) \wedge x \geq 0. \tag{14}$$

$$iposs(Find, s) \equiv holds(pos(x), s) \wedge x \geq 0. \tag{15}$$

Effect axioms

$$[legal(do(\langle MoveLeft, d\rangle, s)) \wedge holds(pos(x), s) \wedge n > 0] \supset holds(pos(x - d), do(\langle MoveLeft, d\rangle, s)) \tag{16}$$

$$[legal(do(\langle MoveRight, d\rangle, s)) \wedge holds(pos(x), s) \supset holds(pos(x + d), do(\langle MoveRight, d\rangle, s)). \tag{17}$$

Ramification State Constraints

The definitions of *central situations* and *hawk situations* are given as an equivalence. Also, constraint (20) states that pos is a functional fluent.

$$holds(Center, s) \equiv holds(pos(x), s) \wedge -1 < x < 1 \tag{18}$$

$$holds(Hawk, s) \equiv holds(pos(x), s) \wedge 0 > x \tag{19}$$

$$holds(pos(x), s) \wedge holds(pos(y), s) \supset x = y. \tag{20}$$

Based on the approach presented in [Pinto 1999], the effect axioms and the ramification state constraints

for the fluent *pos* are replaced by *successor state axioms*. These axioms are derived, by syntactic transformations, from the effect axioms and ramification constraints. The following axioms are obtained after some simplifications:

Successor State Axioms:

$$\begin{aligned} &legal(do(a,s)) \supset [holds(pos(x), do(a,s)) \equiv \\ &\quad holds(pos(x+d), s) \wedge a = \langle MoveLeft, d\rangle \vee \\ &\quad holds(pos(x-d), s) \wedge a = \langle MoveRight, d\rangle \vee \\ &\quad holds(pos(x), s) \wedge a = Find]. \end{aligned} \tag{21}$$

Therefore, the frogbot theory is composed by the PSC axioms, along with axioms (8)-(15), (18)-(19) and (21).

Next, we show how to encode the PSC axioms in MATHEMATICA.

REASONING

In this subsection we present an approach for the computational realization of a system that reasons about actions based on theories written in the Probabilistic Situation Calculus. The approach is based on the encoding of the logical theories into rewrite rules of Mathematica [Wolfram 1996]. We consider the frogbot example to illustrate the approach.

Mathematica Encoding of Domain Independent Definitions

In the example, we will use the *uniform* and the *exponential* probability distributions. The uniform distribution has density over all points that distance one or less from a given point s. To specify the distributions, we explicitly encode their *cummulative distribution functions* (cdf) and their inverses (idf). In the examples, we'll use the following:

```
cdfuniform[s_] :=
Function[x,If[x≤s-1,0,
     If[x≥s+1,1,1/2(x-s+1)]]];
idfuniform[s_] := Function[x,s-1+2x];
cdfexp[m_] := Function[x, UnitStep[x]
(1 - E^(-x/m))];
idfexp[m_] := Function[z, m
Log[1/(1-z)]];
```

Note that, in order to uniformly handle discrete and continuous distributions, one can model discrete distributions with real functions by making use of the *Dirac delta*. Recall that the Dirac delta is a function that is zero over $\mathbb{R}$ except in an infinitesimal neighborhood around 0, whose indefinite integral is 1.

The Monte–Carlo simulation for the univariate case is fairly straightforward. Given a fluent, and an input sequence, we want to determine the probability that the fluent will hold after executing the input sequence with the corresponding reactions. Since the reactions are random, the simulation generates the random reaction from the corresponding inverse cumulative probability distribution, this is simply done as:

```
gen[i_,s_] := idf[i,s][Random[]];
```

Thus, given an input and a situation, the gen function gives a random reaction. Notice that the probability distribution of the reaction might depend on the situation where the input is performed. The `idf` function is domain dependent, hence is specified in the domain axiomatization below.

A Monte–Carlo run is defined as a Bernoulli experiment consisting of taking an input sequence and simulating the execution of the ordered sequence of inputs, each followed by its random reaction. We obtain the situation reached in an Monte–Carlo sampling recursively as follows:

```
gensit[{}, s_] := s;
gensit[w_, s_] := gensit[Rest[w],
     do[{First[w],gen[First[w], s]}, s]];
```

The experiment is successful if the fluent of interest (parameter of the experiment) holds over the reached situation.

```
mcrun[f_, w_, s_] := holds[f, gensit[w, s]];
```

As stated in Proposition 1, in order to estimate the probability that fluent `f` will hold after an input sequence `w`, we use a frequency count. Thus, we run the Monte–Carlo experiment n times, and the estimate of the probability that the outcome will be such that `f` holds is the proportion of times that the experiment succeeds. Hence, we define the `prob` function as follows:

```
prob[f_,w_,s_,n_] := N[Count[
    Table[mcrun[f,w,s],{jrun,1,n}],True]/n];
```

Belief is modeled similarly, and hence, the ratio obtained in Proposition 2 can be calculated as follows:

```
ratio[l_,f_,o_,r_]:=
 Apply[Plus,
```

```
  Map[
   Function[s,If[holds[f,s],cdf[o,s]'[r],0]],
  1]]/
 Apply[Plus,
  Map[
   Function[s, cdf[o, s]'[r]],
  1]];

bel[f_,w_,s_,o_,r_,n_] := N[ratio[
     Table[gensit[w,s],{jrun,1,n}],f,o,r]];
```

The domain independent situation calculus axioms required are direct translations of logical axioms introduced before. These are:

```
poss[{i_,r_},s_] := iposs[i,s]∧
cdf[i,s]'[r]>0;
legal[s0] := True;
legal[do[{i_,r_},s_]] :=
legal[s]∧poss[{i,r},s];
holds[f_,do[a_,s_]] :=
holds[f,s]/;¬legal[do[a,s]];
```

Here the string `/;` should be read as `if`.

Frogbot

Now we translate the frogbot example to illustrate the reasoning capabilities of the oracle in a simple domain in which the probabilities have a continuous distribution. Recall that the frogbot can produce two possible proper inputs, `moveleft` and `moveright`, and the observation `find`. When an input is performed, the random reaction obtained is the distance by which it moves. The random reactions are modeled in our oracle as follows:

```
cdf[moveleft,s_] := cdfexp[3] ;
idf[moveleft,s_] := idfexp[3] ;
cdf[moveright,s_] := cdfexp[2] ;
idf[moveright,s_] := idfexp[2] ;
cdf[find, s_] := cdfuniform[pos[s]];
idf[find, s_] := idfuniform[pos[s]];
```

Notice that, we specify the cumulative distribution function and its inverse. The latter is necessary for performing the simulation in the Monte-Carlo approach to estimate probabilities of fluents holding.

We will make use of the following auxiliary definition:

```
pos[s_]:=iota[x,holds[position[x],s]];
obs[s_]:=iota[x,holds[observation[x],s]];
```

this means that `pos` is a functional fluent, whose value corresponds to the position of the frogbot in a given situation. It is assumed that all observations are functional, and so a similar auxiliar definition can be applied to `obs`. As explained before, the function `iota` gives, in this case, the value of `x` such that `x` is the position of the frogbot in the specified situation. For these definitions, we appeal to the `iota` function. In the `pos` case, `iota` gives the value of `x` such that `holds[position[x],s]` is true. This value has to be unique, in the implementation it gives the first solution it encounters, if any. The Mathematica definition is:

```
iota := Function[z,c,Solve[c,z][[1,1,2]]];
```

Next, we specify the preconditions for the inputs and the initial situation

```
iposs[i_,s_]:=(i===moveright ∨
   i===moveleft ∨ i===find) ∧ pos[s]≥0;
holds[position[x_],s0]:=(x==0);
holds[observations[x_],s0]:=(x==0);
```

The successor state axioms for the position fluent in this domain is as follows:

```
holds[position[x_],do[{i_,y_},s_]] :=
  ((i===moveleft ∧ holds[position[x+y],s]) ∨
  (i===moveright ∧ holds[position[x-y],s]) ∨
  (i=!=moveleft ∧ i=!=moveright ∧
   holds[position[x],s]))/;legal[do[{i,y},s]];

holds[observation[x_],do[{i_, y_},s_]] :=
       ((i === find ∧ x == y)
         ∨ (i =!= find ∧
              x == 0)) /; legal[do[i, y, s]];
```

In this example, we define two fluents in terms of the position of the walking man. First, `hawk` is a fluent that tells us whether or not the walking man is to the right of the initial position. Also, the fluent `center` holds whenever the position of the walking man is within one unit from the center. Notice that these two fluents are acting as abbreviations for their corresponding definitions.

```
holds[hawk,s_]:=pos[s]≥0;
holds[center,s_]:=pos[s]≥-1∧pos[s]≤+1;
```

Given the above domain specification, along with the general domain independent specification of section , we can query our oracle. First, we ask general situation calculus questions that take a specific situation (built with specific action executions, always starting at s0) and asking various properties of such a situation:

```
legal[do[{moveright,7},s0]]
```

```
True
holds[position[x],do[{moveright,7},s0]]
-7+x==0
holds[hawk,do[{moveright,7},
              do[{moveleft,2},s0]]]
False
holds[hawk,do[{moveright,7},
              do[{moveright,2},s0]]]
True
holds[center,do[{moveleft,1},s0]]
True
```

Now, we estimate the real probabilities of fluents hawk and center holding at the specified situations, by using a Monte–Carlo approach with 1000 points. Belief updates for these fluents are also computed. It is easy to verify that the results are very close to the theoretical results expected.

```
prob[hawk,{moveright},s0,1000]
1.
prob[hawk,{moveleft},s0,1000]
0.
prob[hawk,{moveright,moveleft},s0,1000]
0.41
prob[center,{moveleft},s0,1000]
0.3
prob[center,{moveleft,moveright},s0,1000]
0.299
bel[center,{moveright},s0,find,1.8,1000]
0.161663
bel[center,{moveright},s0,find,0.5,1000]
0.704174
bel[hawk,{moveright,moveleft},s0,
                        find,0,1000]
0.482428
```

CONCLUSIONS

With this paper we continued our work on the development of a logical language to support Knowledge Representation and Reasoning for dynamic domains that have uncertain actions. The main contribution of the research reported here (which should be considered upon those already established in [Mateus et al 2001]) is the effective treatment of *probabilistic temporal projection* and *probabilistic update belief* on agents having discrete, continuous or mixed probability distributions modeling their uncertain actions. In detail, we consider an extension of the Situation Calculus, which we call the Probabilistic Situation Calculus (PSC), and endow this logic with terms to deal with the probabilities related with beliefs. This language will be the ground to specifying an agent that executes uncertain actions. Since PSC is a first order logic, first-order reasoning is inherited and it is shown to be sound to a given semantics.

To deal with probabilistic reasoning we provide a Monte-Carlo algorithm as an efective interpretation of the terms dealing with beliefs, all other reasoning on real numbers relies on an oracle, that we implement using MATHEMATICA. Probabilistic Situation Calculus specifications can be translated to a conditional rewriting system, where it is also possible to encode the given Monte-Carlo algorithms. With the frogbot example, we were able to give an idea of the reasoning capabilities of the rewriting system of MATHEMATICA endowed with our algorithms. A detailed comparison with related works, as well as the treatment of observation planning and rewards is also to be included in an extended version of this paper.

Acknowledgements

Paulo Mateus was partially supported by FCT grant SFRH/BPD/5625/2001 and FCT project POCTI/-2001/MAT/37239. António Pacheco was partially supported by FCT grant SFRH/BSAB/251/01 and FCT projects POSI/34826/CPS/2000 and POSI/-42069/CPS/2001.

References

F. Bacchus, J. Halpern, and H. Levesque. Reasoning about noisy sensors and effectors in the Situation Calculus. *Artificial Intelligence*, 111(1-2):171–208, 1999.

C. Baral and L. Tuan. Reasoning about Actions in a Probabilistic Setting. In *Commonsense 2001, Fifth Symposium on Logical Formalizations of Commonsense Reasoning*, 2001. URL = http://www.-cs.nyu.edu/faculty/davise/commonsense01/.

M. Evans and T. Swartz. *Approximating integrals via Monte Carlo and deterministic methods.* Oxford University Press, Oxford, 2000.

H. Grosskreutz and G. Lakemeyer. Turning High-Level Plans into Robot Programs in Uncertain Domains. In *ECAI*, 2000.

H. Grosskreutz and G. Lakemeyer. Belief Update in the pGOLOG Framework. In *Commonsense 2001, Fifth Symposium on Logical Formalizations of Commonsense Reasoning*, 2001. URL = http://www.-cs.nyu.edu/faculty/davise/commonsense01/.

P. Mateus, A. Pacheco, J. Pinto, A. Sernadas, and C. Sernadas. Probabilistic situation calculus. *Annals of Mathematics and Artificial Intelligence*,

32(1/4):393–431, 2001.

J. Pinto. *Temporal Reasoning in the Situation Calculus.* PhD thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, February 1994. URL = `ftp://ftp.cs.toronto.edu/~cogrob/jpThesis.ps.Z`.

J. Pinto. Compiling ramification constraints into effect axioms. *Computational Intelligence*, 15(3):280–307, 1999.

J. Pinto, A. Sernadas, C. Sernadas, and P. Mateus. Non-determinism and uncertainty in the Situation Calculus. In *Proceedings of the 12th International Florida AI Research Symposium, FLAIRS'99*, pages 454–460. AAAI Press, 1999.

J. Pinto, A. Sernadas, C. Sernadas, and P. Mateus. Non-determinism and uncertainty in the Situation Calculus. *International Journal of Uncertainty, Fuzziness and Knowledge Based Systems*, 8(2), 2000.

D. Poole. Decision theory, the Situation Calculus, and conditional plans. *Linköping Electronic Articles in Computer and Information Science*, 3(8), 1998. URL = `http://www.ep.liu.se/ea/cis/1998/008/`.

R. Reiter. *The Frame Problem in the Situation Calculus: A Simple Solution (sometimes) and a completeness result for goal regression*, pages 359–380. Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy. Academic Press, San Diego, CA, 1991.

R. Reiter. Knowledge in action: Logical foundations for describing and implementing dynamical systems. Book Draft, available from `http://www.cs.utoronto.ca/cogrobo`, 2000.

S. Resnick. *A Probability Path.* Birkhäuser Boston Inc., Boston, MA, 1999.

S. Wolfram. *The Mathematica Book.* Wolfram Media, 3 edition, 1996.

Merging and Integration II

A Model-Based Diagnosis Framework for Distributed Embedded Systems*

Gregory Provan
Rockwell Scientific Company,
1049 Camino Dos Rios, Thousand Oaks, CA 91360
gprovan@rwsc.com

Abstract

We present a distributed model-based diagnostics architecture for embedded diagnostics. We extend the traditional model-based definition of diagnosis to a distributed diagnosis definition, in which we have a collection of distributed sub-systems whose interconnectivity is described by a directed graph. Assuming that each sub-system can compute a local minimal diagnosis based only on sensors internal to that sub-system and knowledge only of its own system description, we describe an algorithm that guarantees a globally sound, complete and minimal diagnosis for the complete system. By compiling diagnoses for groups of sub-systems based on the interconnectivity graph, the algorithm efficiently synthesizes the local diagnoses into a globally-sound system diagnosis.

1 INTRODUCTION

This article proposes a new technique for diagnosing distributed systems using a model-based approach. We assume that we have a system consisting of a set of inter-connected sub-systems, each of which computes a local (sub-system) diagnosis. Each sub-system can consist of anything from a single component to a large set of interconnected components. We extend the structure-based diagnosis framework of Darwiche (Darwiche, 1998) to a distributed approach for synthesizing sub-system diagnoses into a globally-sound system diagnosis. Unlike previous approaches that compute diagnoses using the system observations and a component-level system description (Darwiche, 1998; Deb *et al.*, 1998; de Kleer and Williams, 1987), we can take advantage of more abstract system descriptions (e.g., descriptions of sub-systems based on a hierarchical specification), and adopt a diagnosis synthesis process that operates in the space of minimal diagnoses. Assuming that each sub-system can compute a local minimal diagnosis based only on sensors internal to that sub-system and knowledge only of the component sub-system description, we describe an algorithm that guarantees a globally sound, complete and minimal diagnosis for the complete system.

This algorithm uses as input the directed graph (digraph) describing the connectivity of distributed subsystems, with arc directionality derived from the causal relations between the sub-systems. Given that such real-world graphs are either tree-structured or can be converted to tree-structured graphs, we propose a tree-based message-passing algorithm which passes diagnoses as messages and synthesizes local diagnoses into a globally minimal diagnosis. This approach can be applied to systems with arbitrary topologies through transforming sub-system graphs with arbitrary topologies into directed trees. By compiling diagnoses for collections of components (as determined by the graph's topology), we can improve the performance of distributed embedded systems.

One important point to stress is that this approach synthesizes diagnoses computed locally, and places no restriction on the technique used to compute each local diagnosis (e.g., neural network, Bayesian network, ATMS), provided that each local diagnosis is a least-cost or most-likely diagnosis.

The approach presented in this article assumes that all faults are diagnosable (i.e., can be isolated) through a centralized algorithm. We examine whether a distributed approach can diagnose all faults, since a distributed algorithm can isolate faults no better than a

Research supported in part by The Office of Naval Research under contract number N00014-98-3-0012.

centralized algorithm. Issues relating to restricted diagnosability of both centralized and distributed algorithms due to insufficient observable data (e.g., when the suite of sensors is insufficient to guarantee complete diagnosability) are examined in (Provan, 2002).

This article is organized as follows. Section 2 introduces the application model that we use to demonstrate our approach. Section 3 introduces our modeling formalism, and specifies our centralized and distributed model. Section 4 describes how we diagnose distributed models, and Section 5 describes how to convert directed graphs with arbitrary topologies into directed trees. Section 6 surveys some related work on this topic. Section 7 summarizes our conclusions.

2 IN-FLIGHT ENTERTAINMENT EXAMPLE

Throughout this article we use a simplified example of an In-Flight Entertainment (IFE) system. An IFE system provides audio, video and game entertainment on board passenger aircraft. This system has a main module that accepts passenger requests and generates audio/video/game signals, which are then transmitted (via RF signals) to passengers through several intermediate components. Figure 1 shows the schematic for an IFE system fragment where we have (1) a transmitter module (Tx) that generates 10 movie channels (consisting of both video and audio signals) and 10 audio channels; (2) two area distribution boxes (ADB); and (3) attached to each ADB_i we have two passenger units, P_{i1} and P_{i2}. For ADB j, passenger i, $i = 1, 2$ has a controller C_{ji} for selecting a video or audio channel, plus an audio unit α_i and video display υ_i. Control signal C_{ji} is sent by passenger i to ADB_j and then to the transmitter, which in turn sends an RF signal (RF) to each passenger.

We adopt a notion of causal influence for describing how different components affect the value of a signal as it propagates through the system. For example, the RF signal causally influences the passenger audio and video outputs. In this model the observables are the control signals, plus for passenger i downstream of ADB_j sound (S_{ji}) and video-display (VD_{ji}). We assign a fault-mode to the transmitter and to each ADB and passenger unit.

Our modeling approach makes the following assumptions. First, we can specify a system using an object-oriented approach. In other words, a system can be defined as a collection of components, which are connected together, e.g., physically, as in an HVAC system, or in terms of data transmission/reception, as in the IFE example. Our primary component consists of a *block*, which has properties: input set, output set, fault-mode, and equations. Given the fault-mode and input set, the equations provide a mapping to the output set. In other words, the inputs are the only nodes with causal arcs into the block, and the outputs are the only nodes with causal arcs out of the block. Typically, we have causal dependence of block outputs ω_i on inputs ℓ_i, *i.e.* $\omega_i \propto \ell_i$.[1]

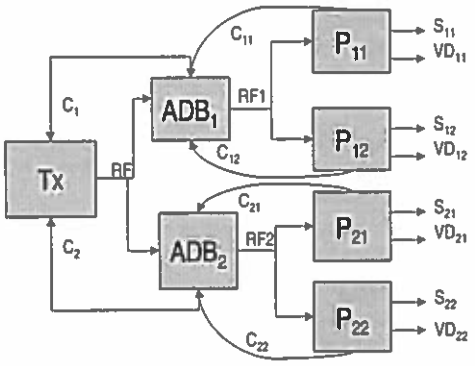


Figure 1: Schematic of IFE fragment, showing the main modules and the directed arcs of data-flows.

This distributed model consists of a set of sub-models, or blocks, which may be connected together. In our IFE example, the transmitter block has inputs of control signals C_1 and C_2, and output an RF signal.

Second, we assume that each component computes diagnoses based on data local to the component. We do not place any restrictions on the type of algorithm used to compute the diagnosis, except that the diagnosis be a least-cost diagnosis. We will describe the cost function used by our synthesis algorithm in the following section.

3 MODEL-BASED DIAGNOSTICS USING CAUSAL NETWORKS

This section formalizes our modeling and inference approach to diagnostics and control reconfiguration. We first introduce the model-based formalism, and then extend these notions to capture a distributed model-based formalism.

[1]The causal function $\propto$ can be be generalized to include propositions, relations, probabilistic functions, qualitative differential equations, etc. We don't address such a generalization here.

3.1 FLAT (CENTRALIZED) MODELS

We adopt and extend the model-based diagnosis representation of Darwiche (Darwiche, 1998). We model the system using a causal network:

Definition 1 *A system description is a four-tuple* $\Phi = (\mathcal{V}, \mathcal{G}, \Sigma)$, *where*

- $\mathcal{V}$ *is a set of variables comprising two variable types:* $\mathcal{A}$ *is a set of variables (called assumables) representing the failure modes of the components,* $\mathbf{V}$ *is a set of non-assumable variables* ($\mathbf{V} \cap \mathcal{A} = \emptyset$) *representing system properties other than failure modes;*
- $\mathcal{G}$ *is a directed acyclic graph (DAG) called a causal structure whose nodes are members in* $\mathbf{V} \cup \mathcal{A}$ *and whose directed arcs represent causal relations between pairs of nodes;*
- *and* Σ *is a set of propositional sentences (called the domain axioms) constructed from members in* $\mathbf{V} \cup \mathcal{A}$ *based on the topological structure of* $\mathcal{G}$.

This definition of system description differs from the standard definition (called SD in (Reiter, 1987)) only in that we include a graph $\mathcal{G}$ to complement the domain axioms, a set of failure modes (commonly called COMPS) and non-assumable variables. $\mathcal{G}$ imposes additional restrictions on the system description as compared to Reiter's definition, such as that of directed acyclic relations.

The set of non-assumable variables consists of two exclusive subsets: $\mathbf{V}_{obs}$ (the set of observables) and $\mathbf{V}_{unobs}$ (the set of unobservables).

We can capture structural properties of the system description using the directed acyclic graph, or DAG, $\mathcal{G}$.[2] For example, if an actuator determines if a motor is on or not, we say that the actuator causally influences the motor. More generally, A may directly causally influence B if A is a predecessor of B in $\mathcal{G}$. We use $B \propto A$ to denote the direct causal influence of the value of B by the value of A.[3] Through transitivity, we can deduce *indirect* causal influence. For example, if $B \propto A$ and $C \propto B$, then A indirectly influences C.

We capture and exploit the directionality of causal relations during all phases of diagnostic inference. For example, if we have an abstract hierarchical specification of a system and compute diagnostics for each abstract hierarchical block, we still preserve the directionality of causality among the abstract blocks. We exploit this directionality using a diagnostic synthesis algorithm operating on a *directed* tree.

We capture the notion of direct causal influence, i.e., a node N and those nodes that are directly causally affected by N, using a *clan*. We define the notion of the clan of a node N in terms of graphical relationships as follows:

Definition 2 (Clan) *: Given a DAG* $\mathcal{G}$, *the clan* $Y(N_i)$ *of a node* $N_i \in \mathcal{G}$ *consists of the node* N_i *together with its children in* $\mathcal{G}$.

For simplicity of notation, we will denote the clan for node N_i, $Y(N_i)$, as Y_i. We adopt the notion of clan because it facilitates the process of synthesizing diagnoses computed at a set of distributed nodes organized in a tree structure. The intuition is as follows. Given a tree of depth 1[4], e.g., a parent node N with child-nodes C_1, C_2 and C_3, a minimal diagnosis can be computed trivially, based on the parent/children structure. By decomposing an arbitrary tree into a collection of depth-1 sub-trees, we can compute a global minimal diagnosis by recursively computing the minimal diagnosis in each sub-tree. Each sub-tree has the structure of a clan.

This approach bears some resemblance to techniques that use clique-tree decompositions generated based on graphical relations known as a family, e.g., (Fattah and Dechter, 1995).[5] We discuss these relationships in Section 6.

It is also important to define instantations of subsets of observables:

Definition 3 (Restriction) *We denote by* θ_i *the restriction of an instantiation* θ *of variables* V *to the instantiation of a subset* V_i *of* V. *We denote the restriction of variable set* T *to variables in sub-system description* Φ_i *by* T^{Φ_i}.

One of the key elements of diagnosing a system is the instantiation of observables, since a diagnosis is computed for abnormal observable instantiations.

Definition 4 (Instantiation) θ^{Φ_i} *is an instantiation of observables* ${V_{obs}}^{\Phi_i}$ *for system description* Φ_i.

[2] In other system description specifications, e.g. (Dressler and Struss, 1996), these structural relations are captured using logical sentences.

[3] This notion of causal influence does not guarantee that A influences B, but that A *may* influence B.

[4] The depth of a tree is the length of longest path from root to leaf of the tree.

[5] A *family* is defined as a node together with its parents in $\mathcal{G}$.

Θ^{Φ_i} *denotes the set of all instantiations of observables* ${V_{obs}}^{\Phi_i}$.

We specify failure-mode instantiations and partition the possible states into normal states and faulty states as follows:

Definition 5 (Mode-Instantiation) $\mathcal{A}^*$ *is an instantiation of behavior modes for mode-set* $\mathcal{A}$. *Further, we decompose* $\mathcal{A}^*$ *such that* $\mathcal{A}^* = \mathcal{A}^F \cup \mathcal{A}^\emptyset$, *where* $\mathcal{A}^\emptyset$ *denotes normal system behaviour, i.e. all modes are normal, and* $\mathcal{A}^F$ *denotes a system fault, which may consist of simultaneous faults in multiple components.*

An assumable (behavior-mode variable) specifies the discrete set of behavior-states that a component can have, e.g., an AND-gate can be either *OK*, *stuck-at-0*, or *stuck-at-1*. Our IFE-system, with component-set $\{Tx, ABD_1, ADB_2, P_{11}, P_{12}, P_{21}, P_{22}\}$, can have a mode-instantiation in which all components are OK except P_{11}, which is in *audio-fail* mode. In this case we have $\mathcal{A}^\emptyset = \{Tx - mode = OK, ABD_1 - mode = OK, ADB_2 - mode = OK, P_{12} - mode = OK, P_{21} - mode = OK, P_{22} - mode = OK\}$ and $\mathcal{A}^F = \{P_{11} - mode = audio\text{-}fail\}$.

3.2 DISTRIBUTED SYSTEM DESCRIPTIONS

This section describes our distributed formalism, which applies to collections of interconnected components, or blocks. We assume that a distributed system description is provided either by the user or is deduced from the physical constraints of available local diagnostic agents and physical connectivity. For example, many engineering systems, such as commercial aircraft, are subdivided into Line-Replaceable Units (LRUs), based on factors such as fault-isolation capabilities, physical constraints, and ease of repair. An LRU typically consists of a number of connected subsystems, as in the Passenger Unit of the IFE example, which consists of circuit-cards to select audio/video channels and to drive the audio and video output devices. It is standard practice in commercial aircraft to isolate faults only to the LRU-level, and replace faulty components only at the LRU-level.

Definition 6 (Decomposition Function) *a decomposition function is a mapping* $\psi(\Phi) = \Phi_{dist}$ *that decomposes a centralized system description* Φ *into a distributed system description* $\Phi_{dist} = \{\Phi_1, ..., \Phi_m\}$. *The distributed system description induced by a decomposition function* ψ *is defined by a decomposition* Π *over the system variables* $\mathcal{V}$, *i.e. a collection* $\mathcal{X} = \{X_1, ..., X_m\}$ *of nonempty subsets of* $\mathcal{V}$ *such that (1)* $\forall i = 1, ..., m,\ X_i \in 2^{\mathcal{V}}$; *(2)* $\mathcal{V} = \cup_i (X_i | X_i \in \Pi)$. *When* $\xi_{ij} = X_i \cap X_j \neq \emptyset$, *we call* ξ_{ij} *the separating set, or common set, of variables between* Φ_i *and* Φ_j.

We can describe a distributed system description in terms of a *decomposition graph*. A decomposition graph is a graphical representation of the system model, when viewed as a collection of connected blocks. In this graph each vertex corresponds to a block, and each directed edge corresponds to a directed (causal) link between two blocks. Figure 2 shows the decomposition graph for the extended IFE example.[6]

A *decomposition graph* is a directed tree, or D-tree, which is defined as follows:

Definition 7 *A D-tree* $\mathcal{T}_\mathcal{D}$ *is a directed graph with vertices* $\mathcal{V}_{\mathcal{T}_\mathcal{D}}$ *and a vertex* V_0, *called the root, with the property that for every vertex* $V \in \mathcal{V}_{\mathcal{T}_\mathcal{D}}$ *there is a unique directed walk from* V_0 *to* V.

Our approach uses two different D-trees, a system decomposition graph and a clan graph.

Definition 8 *A system decomposition graph* $G_\mathcal{X}$ *is an edge-labeled D-tree* $G(\mathcal{X}, \mathcal{E}, \xi)$ *with (1) vertices* $\mathcal{X} = \{X_1, ..., X_m\}$, *where each vertex consists of a collection of variables of* $\mathcal{G}$, *(2) directed edges join pairs of vertices with non-empty intersections, and arc direction is specified by the causal direction of the arcs between blocks in the decomposition graph, i.e.,* $\mathcal{E} = \{(X_j, X_k) | X_i \cap X_j \neq \emptyset,\ X_k \propto X_j\}$, *and (3) edge labels (or separators) defined by the edge intersections,* $\xi = \{\xi_{ij} | X_i \cap X_j \neq \emptyset\}$.

We assume that in a distributed system description, for any block all sensor data is local, and all equations describing distributed subsystems refer to local sensor data and local conditions.

3.3 DIAGNOSIS SPECIFICATION

We adopt the standard notion of diagnosis (Reiter, 1987) as follows:

Definition 9 (Diagnosis) *Given a system description* Φ *with system axioms* Σ *and an instantiation* θ *of* $\mathbf{V}_{obs}$, *a diagnosis* $D(\theta)$ *is an instantiation of behavior modes* $\mathcal{A}^F \cup \mathcal{A}^\emptyset$ *such that* $\Sigma \cup \theta \cup \mathcal{A}^F \cup \mathcal{A}^\emptyset \not\models \bot$.

This diagnostic framework provides the capability to rank diagnoses using a likelihood weight κ_i assigned to

[6] We do not show the feedback loops of control requests $(C_1, C_2, C_{11}..., C_{22})$ since all edges concerning observables can be cut (Darwiche and Provan, 1996).

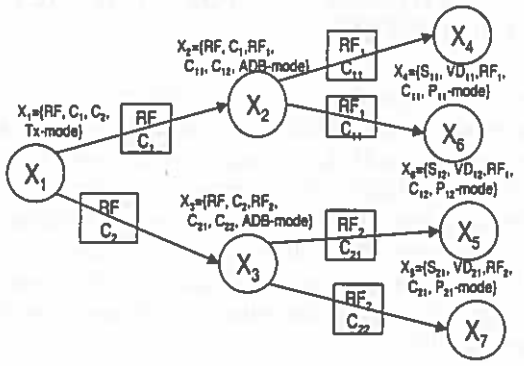


Figure 2: Decomposition graph of extended IFE system description. Here an oval corresponds to a vertex, and a block corresponds to a sepset. We specify the variables associated with each vertex in the graph.

each assumable $\mathcal{A}_i$, $i = 1, ..., m$. Using the likelihood algebra defined in (Darwiche, 1998), we can compute the likelihood assigned to each diagnosis for observation θ. We refer to a (diagnosis, weight) pair using $(D(\theta), \kappa)$. We use the weights to rank diagnoses, i.e., least-weight diagnoses are the most-likely. This provides a notion of *minimal diagnosis*, i.e. a diagnosis of weight κ such that there exists no lesser-weight diagnosis.

3.4 LOCAL/GLOBAL DIAGNOSTICS

Our methodology rests on the determination of when component diagnoses are independent, in which case the global diagnosis is just the conjunction of the component diagnoses. We apply the decomposition theorem of (Darwiche, 1998) to this case of distributed diagnostics:

Theorem 1 *If we have a system description Φ consisting of two component system descriptions Φ_1 and Φ_2, and a system observation θ, if the variables shared by Φ_1 and Φ_2 all appear in θ, then*

$$D^{\Phi}(\theta) \equiv D^{\Phi_1}(\theta_1) \wedge D^{\Phi_2}(\theta_2).$$

This theorem states that a diagnosis is decomposable provided that the system observation contains the variables shared between Φ_1 and Φ_2. However, what happens when the observation θ does not contain all variables shared between Φ_1 and Φ_2? One solution (Darwiche, 1998) is to decompose the computation of D^{Φ} by performing a case-analysis of all shared variables ξ_{12}. However, this case-analysis approach is exponential in $|\xi_{12}|$, the number of variables on which we do case-analysis. Hence if we wanted to embed the diagnostics code, such a case-analysis might be too time-consuming when performed on a system-level model.

In the following we assume that each component computes a *local diagnosis*, i.e., a diagnosis based only on local observables and on equations containing only local variables. In contrast a *global diagnosis* is one based on global observables and on equations describing all system variables. Our task is to integrate these local component diagnoses into a globally sound, minimal and consistent diagnosis, since for many systems the diagnostics generated locally are either incomplete or not minimal.

Note that we can obtain global diagnostics for a modular system by composing local blocks and diagnosing the entire system model. However, it is true in many cases that global and local diagnostics may differ. We now define a notion of correspondence between local and global diagnoses.

The conjunction of the set of distributed system descriptions is defined as $D_{dist}(\theta) = \bigwedge_{\Phi_k \in B} D^{\Phi_k}(\theta)$, and we know that $D_{dist}(\theta) = D(\theta)$ only when $\theta \equiv \bigcup_{i,j} \xi_{ij}$.

We can compute the diagnoses for this set of distributed system descriptions either using an on-line algorithm, or by pre-computing the set of diagnoses for $D_{dist}(\theta)$. In the following, we outline the compiled method of diagnosis.

We define a table, called a clan table, to specify local and global diagnoses for collections of blocks. This table compiles the local case-analysis required by Theorem 1. We will show later how to use this table for our diagnosis synthesis algorithm.

Definition 10 *A* **clan (or local/global diagnosis) table** *for block-set $B = \{\Phi_i, ...\Phi_j\}$ is a table consisting of tuples (observable-intantiation, global diagnosis, weight) for all abnormal instantiations of observables θ in B.*

Note that we can use the compositionality of blocks to show that any time we compose a system description from multiple blocks, we obtain "global" diagnostics for that composed system description when we compute diagnoses over the composed system description. Hence the "global" diagnosis for each collection of blocks is computed from a system description generated from the composition of the system descriptions of the blocks in B, using the observables from B.

Example 1 Table 1 contrasts the local and global diagnoses for a set of scenarios where the set B of blocks is an ADB with downstream passenger units. In these

scenarios, we compute the (probabilistically) most-likely diagnosis, assuming that all faults are equally likely, i.e., have weight 1. Moreover, in defining a local diagnosis in Table 1, we report the conjunction of all local diagnoses, i.e. the local diagnosis is *ADB-diagnosis* $\wedge$ P_1*-diagnosis* $\wedge$ P_1*-diagnosis*. In scenarios 1, 2 and 4, the local and global diagnoses are identical. However, in scenarios 3, 5 and 6, they differ: the passenger units each assume a local fault, whereas the transmitter unit is the faulty one (since a single transmitter fault is much more likely the two simultaneous faults, one in each passenger unit).[7]

Given this potential for discrepancy between local and global diagnoses, we map the decomposition graph into a representation, the clan graph, from which we can synthesize globally sound and complete minimal diagnoses from local minimal diagnoses. A clan graph has as its nodes collections of blocks, where each collection consists of a block and its children in $\mathcal{G}_\mathcal{X}$. Figure 3 shows the clan graph for the extended IFE example.

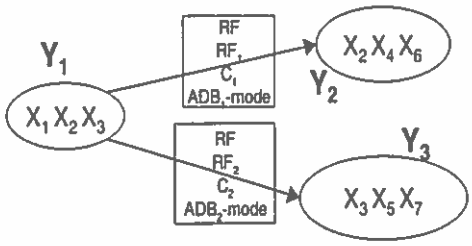


Figure 3: Clan graph of extended IFE system description.

Definition 11 (Clan graph) *: A clan graph $G_\mathcal{Y}$ of a DAG $\mathcal{G}(V,E)$ of vertices V and edges E is an edge-labeled D-tree $G(\mathcal{Y},\mathcal{E},\xi)$ defined as follows: (1) vertices $\mathcal{Y} = \{Y_1,...,Y_m\}$, where each node Y_i consists of a clan of $\mathcal{G}$; (2) edges defined by non-empty intersections between pairs of vertices $\mathcal{E} = \{(Y_j,Y_k)|Y_i \cap Y_j \neq \emptyset\}$; and (3) separators defined by the edge intersections $\xi = \{\xi_{ij} = Y_i \cap Y_j\}$.*

The following section shows how we use the clan graph for distributed diagnosis.

[7]These differences arise due to different instantiations of the RF signal in the local and global diagnosis. We hide the details of the case-analysis of shared variables for simplicity of presentation.

4 DISTRIBUTED MODEL-BASED DIAGNOSIS

This section describes our distributed model-based diagnosis algorithm. This algorithm provides a pre-compiled approach that is significantly faster for embedded computation. Our approach uses the structure of the component interconnections to map minimal diagnoses for components first to minimal clan diagnoses and then to minimal system diagnoses. The decomposition graph specifies the subsets of components that share variables.

4.1 DIAGNOSIS OF TREE-STRUCTURED SYSTEMS

We now describe an approach to diagnosing systems with tree-structured decomposition graphs. We later show how this can be generalized such that arbitrary graph topologies can be converted to trees.

We assume that:

- We are provided with the component system descriptions and their connectivity;
- There is a single root in the decomposition graph (which is a component with no parent-components), and each leaf is a component with no child-component;
- Nodes have indices starting at the root (X_1), increasing based on a breadth-first expansion from the root and ending at the $s+1$ leaves, labeled $X_{n-s},...,X_n$;
- Each component computes a local diagnosis based on local observables.

We base our approach on synthesizing diagnoses, starting from the leaf components and ending up at the root of the tree. We first decompose the decomposition graph into a clan graph. Based on the clan graph we construct a clan table for each node in the graph.

Under this scheme, we pre-compute clan tables for each clan in $\mathcal{G}_\mathcal{Y}$. Given an observation θ for blocks $X_i,...,X_k$, where $X_i,...,X_k$ are members of a clan $Y \in \mathcal{G}_\mathcal{Y}$, each block computes diagnostics locally. We then compute the most likely fault-mode assignment for Y through a process we call *diagnostics synthesis*, which entails table-lookup in the clan table of the minimal diagnosis given θ. The algorithm synthesizes final diagnoses, going from the leaves to the root. This guarantees a sound, complete and globally minimum system diagnosis.

Scenario	ADB_1 Unit		Pass. $Unit_{11}$		Pass. $Unit_{12}$		Diagnosis	
	C_{11}	C_{12}	S_{11}	VD_{11}	S_{12}	VD_{12}	LOCAL	GLOBAL
1	audio	audio	nom.	none	nom.	none	–	–
2	audio	audio	**none**	none	nom.	none	P_{11}-*audio-fail*	P_{11}-*audio-fail*
3	audio	audio	**none**	none	**none**	none	P_{11}-*audio-fail*$\wedge$ P_{12}-*audio-fail*	*Xaudio*
4	video	video	nom.	nom.	nom.	**none**	P_{12}-*video-fail*	P_{12}-*video-fail*
5	video	video	nom.	**none**	nom.	**none**	P_{11}-*video-fail*$\wedge$ P_{12}-*video-fail*	*Xvideo*
6	audio	video	**none**	none	**none.**	**none**	P_{11}-*audio-fail*$\wedge$ P_{12}-*video-fail*	ADB_1-*fail*

Table 1: Diagnostic Scenarios. We denote a nominal passenger output of nominal using nom., and abnormal observable data in bold-face. *Xaudio* denotes degraded audio, and *Xvideo* denotes degrated video.

In this approach we first need to pre-compute the clan table, and then use that table for diagnostic synthesis. We can pre-compute the clan table from a set of blocks $\{\Phi_1, ..., \Phi_k\}$ as follows:

1. Generate the decomposition graph $G_{\mathcal{X}}$ from $\{\Phi_1, ..., \Phi_k\}$, with indices increasing in a breadth-first manner from the root.
2. Generate the clan graph $G_{\mathcal{Y}}$ of $G_{\mathcal{X}}$.
3. Compute the clan table for each clan Y_i in $G_{\mathcal{Y}}$.

Given an observation θ, the diagnostic synthesis algorithm is as follows:

1. Given observation θ, each block B_i computes its local diagnosis $D^{\Phi_i}(\theta)$ and likelihood $\kappa(D^{\Phi_i})$.
2. Mark all nodes X_i, $i = 1, ..., n$ with flag=0;
3. Loop for $j = n$ to 1:
   - If flag=0 for X_j do:
     For each node X_i in the clan $Y(X_j)$, look up corresponding clan diagnosis $D^{\Phi_Y}(\theta)$ and weight $\kappa(D^{\Phi_Y}(\theta))$ in the clan-table;

$$\text{If } \kappa(D^{\Phi_Y}(\theta)) < \sum_{k:\Phi_k \in Y} \kappa(D^{\Phi_k}),$$

     - revise fault-mode assignment to nodes in $Y(N_j)$, by (a) setting the minimum-weight diagnosis mode-variable; (b) if any local diagnosis D' is synthesized, update D'.
     - reassign values to variables in Y based on D and θ
     - if reassignment is sound pass message with fault report $D^{\Phi_Y}(\theta)$.
     - Set flag for all $X_i \in Y(X_j)$ to 1;

This algorithm has the following guarantees:

Theorem 2 *Given a tree-structured decomposition graph $\mathcal{G}_{\mathcal{X}}$ and local component diagnoses, diagnostics synthesis will compute a sound and globally consistent set of fault mode assignments for components $\mathcal{X} \in \mathcal{G}_{\mathcal{X}}$ within $O(|\mathcal{Y}|)$ message-passing steps, where $\mathcal{G}_{\mathcal{Y}}$ is the clan graph generated from $\mathcal{G}_{\mathcal{X}}$.*

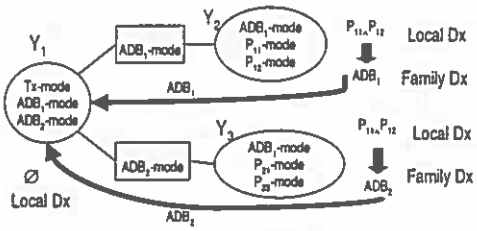


Figure 4: Diagnosis synthesis procedure, Step 1: (a) local diagnoses synthesized at clans, and (b) clan diagnoses are passed between families, as noted by dark arrows.

Example 2 Diagnosis Synthesis in a Clan: Consider Scenario 3 of Table 1. For this observation θ, the total set of possible clan diagnoses is: (P_{11}, *audio-fail*) $\wedge$ (P_{12}, *audio-fail*) $\vee$ (ADB_1, *Xaudio*). The weights of the diagnoses are 2 and 1, respectively.

In computing diagnoses on a purely local basis, the resulting diagnosis is (P_{11}, *audio-fail*) $\wedge$ (P_{12}, *audio-fail*), with weight 2. Note however there is a family diagnosis of weight 1, (ADB_1, *Xaudio*), which is selected since it is of lower weight than the distributed diagnosis. We now instantiate each local component with θ, and set diagnoses as follows: (P_{11}, $\emptyset$), (P_{12}, $\emptyset$), (ADB_1, *Xaudio*). There exists a consistent set of local variable instantantiations for this assignment, so no further message-passing is necessary.

Example 3 Message-Passing: Figure 4 shows the first stage of this procedure. In the graph we show nodes where the variables are restricted to fault mode variables, to simplify the description of message-passing of instantations of mode variables. First, the local diagnoses are computed at each node in the decomposition graph: all four passenger units register a fault, and no other nodes in the decomposition graph register faults. As a shorthand, we denote a fault-weight pair using variable-names for faults, with $\emptyset$ denoting a nominal mode. Then, these faults are synthesized at each clan using the clan-table: fault-weight

pair $(P_{11} \wedge P_{12},\ 2)$ is synthesized into $(ADB_1,\ 1)$, and fault $(P_{21} \wedge P_{22},\ 2)$ is synthesized into $(ADB_2,\ 1)$. Second, the synthesized faults $(ADB_1,\ 1)$ and $(ADB_2,\ 1)$ are sent to the adjacent node in the clan graph, Y_1.

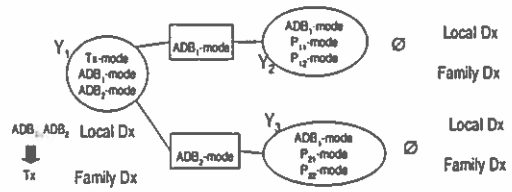


Figure 5: Diagnosis synthesis procedure, Step 2: global diagnoses computed following family diagnosis message-passing.

Figure 5 shows the second stage of this procedure. Fault-weight pair $(ADB_1 \wedge ADB_2,\ 2)$ is synthesized into $(Tx,\ 1)$ at clan Y_1, and all other fault-modes are set to nominal. This is the global minimum-weight fault.

4.2 COMPLEXITY ISSUES

This approach is based on computing diagnoses for the clans of $\mathcal{G}$. Hence, it never needs to diagnose a system description for the entire graph $\mathcal{G}$, but only for the clans of $\mathcal{G}$. As noted in Theorem 2, once the clan tables are computed, given any local component diagnoses, the algorithm is linear in the number of nodes in the clan-graph.

The worst-case complexity of computing a clan table is exponential in the number of variables in the clan table. The memory requirements for storing the clan tables are defined as follows. In the worst case, for a clan with mode variables $\mathcal{A}_1, ..., \mathcal{A}_m$, where each mode variable has $|\omega_{A_i}|$ faulty values, a clan table stores an entry for each of the $\times_i |\omega_{A_i}|$ multiple-fault combinations. For single-fault scenarios, a clan table must store only $\sum_i |\omega_{A_i}|$ entries.

For tree-structured systems the complexity of diagnosing $\mathcal{G}$ is exponential in the clan size, and the complexity is bounded by the largest clan of $\mathcal{G}$. Hence the complexity of initially computing diagnoses is the same for the centralized and distributed approaches. However, for embedded applications, the distributed approach has a complexity advantage, since only clan-table lookup and simple message-passing are required. The major possible drawback is the space complexity of the clan tables. The complexity of logical resolution within a distributed framework have been discussed in detail in (Amir and McIlraith, 2000). The complexity properties are almost identical to those of our approach, even though our task is model-based diagnosis within a directed tree.

5 EXTENSIONS TO NON-TREE-STRUCTURED GRAPHS

This section discusses the applicability of this theory to non-tree-structured graphs, or when a decomposition tree is not provided by the user. We first show the theoretical underpinnings of this applicability, and then discuss experience with real-world system topologies.

5.1 THEORETICAL PERSPECTIVES

The proposed approach has been designed especially for tree-structured systems, when the structure is specified at a fairly abstract (or LRU) level, and not at the most detailed component level. This approach is extensible to arbitrary system topologies, by applying a class of tree-decomposition algorithms that transform arbitrary graphs into trees.

The work on tree-decomposition stems from work on treewidth and graph minors (Robertson and Seymour, 1986). A good review of the literature can be found in (Bodlander, 1997). We define the basic notions below.

Definition 12 *A* tree decomposition *of an undirected graph $G = (V, E)$ is a pair $(\mathcal{X}, T)$ with $T = (I, F)$ a tree, and $\mathcal{X} = \{X_i | i \in I\}$ is a family of subsets of V, one for each node of T, such that*

1. $\bigcup_{i \in I} X_i = V$;
2. *for all edges $\{v, w\} \in E$ there exists an $i \in I$ with $v \in X_i$ and $w \in X_i$, and*
3. *for all $i, j, k \in I$ if j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.*

The *width* of a tree decomposition is $\max_{i \in I} |X_i| - 1$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G.

The treewidth bears close relations to the maximal vertex degree and maximal clique of a graph, so it provides a measure of the complexity of diagnostic inference, among other things. If a graph has a low treewidth then inference on the graph is guaranteed to be easy.

The task of computing treewidth is NP-hard (Arnborg *et al.*, 1987). Many algorithms exist that, given a graph with n variables, will compute an optimal treewidth in time polynomial in n but exponential in the treewidth k; see, for example, (Bodlaender, 1996).

The difference between the standard literature on tree-decompositions and the task addressed here is that the standard literature focuses on undirected graphs, and we focus on directed graphs. The tree-decomposition results have been generalized to directed graphs in (Johnson *et al.*, 2002), and we make use of some of those results here. The key change is that we need to preserve ordering of edges during the decomposition process. To capture such properties, we first need to define a notion of variable ordering, called Z-normality.

Definition 13 *Let $\mathcal{G}$ be a digraph and let $Z \subseteq \mathcal{V}$. A set S is Z-normal if and only if the vertex-sets of the strong components of $\mathcal{G} \setminus Z$ can be numbered $S_1, S_2, ..., S_d$ such that*

1. *if $1 \leq i \leq j \leq d$, then no edge of $\mathcal{G}$ has a head in S_i and tail in S_j, and*
2. *either $S = \emptyset$ or $S = S_i \cup S_{i+1} \cdots \cup S_j$ for some integers i, j with $1 \leq i \leq j \leq d$.*

Definition 14 *A* D-tree decomposition *of a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a pair $(\mathcal{X}, \mathcal{T}_\mathcal{D})$ with $\mathcal{T}_\mathcal{D} = (\mathcal{I}, \mathcal{F})$ a D-tree, and $\mathcal{X} = \{X_i | i \in \mathcal{I}\}$ is a family of subsets of $\mathcal{V}$, one for each node of $\mathcal{T}_\mathcal{D}$, and the edges are numbered $\mathcal{J} = \{1, ..., l\}$ with $\mathcal{F} = \{F_j : j \in \mathcal{J}\}$, such that*

1. *$\bigcup_{i \in \mathcal{I}} X_i = \mathcal{V}$;*
2. *for all edges $\{v, w\} \in \mathcal{E}$ there exists an $i \in \mathcal{I}$ with $v \in X_i$ and $w \in X_i$, and*
3. *for all $i, j, k \in \mathcal{I}$ if j is on the path from i to k in $\mathcal{T}_\mathcal{D}$, then $X_i \cap X_k \subseteq X_j$;*
4. *if $j \in \mathcal{J}$, then $\bigcup_i \{X_i : i \in \mathcal{I}, i > j\}$ is X_j-normal.*

The *width* of a tree decomposition is the least integer w such that for all $i \in \mathcal{I}$, $|X_i \cup \bigcup X_j| \leq w + 1$, where the union is taken over all edges $j \in \mathcal{J}$ incident with i. $\max_{i \in \mathcal{I}} |X_i| - 1$. The *treewidth* of a graph $\mathcal{G}$ is the least integer w such that $\mathcal{G}$ has a D-tree-decomposition of width w.

For the class of applications addressed in this article, the input graphs $\mathcal{G}$ for the system description are digraphs, and the decomposition graph and clan graph are both D-tree decompositions of $\mathcal{G}$. For more general digraph topologies, by applying an algorithm for generating D-tree decompositions, we can convert the digraphs into a decomposition graph, and apply the diagnostic synthesis approach. Many of the properties of undirected tree-decompositions hold for the directed case (Johnson *et al.*, 2002).

5.2 PRACTICAL EXPERIENCE WITH REAL-WORLD SYSTEMS

In our experience in building diagnostic models, many real-world systems have tree-structured abstract system descriptions (decomposition graphs). For example, models from the class of fluid-flow systems (including HVAC systems, pneumatic systems, fuel systems, chemical processing systems, etc.) have either a simple tree or path structure, or they contain loops which can be converted into paths given a sensor positioned on the loop.[8]

For systems without a natural tree-structured decomposition graph, it is possible to convert the system digraph into a decomposition graph using D-tree-decomposition algorithms; however, the treewidth of the resulting decomposition graph would determine the efficiency of the diagnostic synthesis approach. Hence generating a low treewidth decomposition graph for a particular system is topology-dependent. To date, our experiments on simple real-world systems have shown that the resultant decomposition graphs have low treewidth. Further work is necessary to determine how this scales up to large systems.

6 RELATED WORK

Our approach to distributed diagnosis has been preceded by many pieces of related work, and we review several here. Note that is review examines the most relevant work, and does not claim to be exhaustive.

One of the most closely-related pieces of work describes techniques for distributed logical inference (Amir and McIlraith, 2000; McIlraith and Amir, 2001). This work focuses on how to perform logical reasoning and query answering, proposing sound and complete message passing algorithms, by exploiting the tree structure of distributed theories. They examine the complexity of computation, propose specialized algorithms for first-order resolution and focused consequence finding, and propose algorithms for optimally partitioning a theory that is not already distributed. In some ways, our task can be considered a speical case of the general problem that Amir and McIlraith examine. Logical

[8] We showed in (Darwiche and Provan, 1996) that a loop can be broken by the presence of an observable variable in the loop.

inference computes a model, whereas diagnostic inference computes a *minimal* model in the assumables, a subset of the language of the theory. We leverage many aspects of the specific diagnosis problem in our work, that serve to distinguish both our approach and our results. These include the notion of causality, which imposes a directionality on the tree structure and the inference, and the notion of preference. In addition, the task of diagnostic inference depends critically on two classes of distinguished variables, assumables (the literals of interest) and observables (the inputs), and distributed diagnosability depends on how assumables and observables are distributed among the collection of blocks. In addition, if the variables common between two blocks are observable, then from a distributed diagnostics point of view those blocks are independent (Darwiche and Provan, 1996).

The approach presented here bears some relation to diagnostic approaches on trees. Stumptner and Wotawa (Stumptner and Wotawa, 2001) have an algorithm for diagnosing tree-structured systems. This approach assumes a centralized system defined at the component level whereas our approach deals with distributed systems that can be defined at any level of abstraction. In addition, our assumption of sub-systems computing their own diagnoses means that our diagnostic synthesis process is a single-pass algorithm from the leaves of the tree to the root, whereas Stumptner and Wotawa need a two-pass approach since they must first enumerate all component diagnoses. A second major tree-based method uses a clique-tree decomposition of a system, e.g., the diagnostic method of (Fattah and Dechter, 1995). A clique-tree is a representation that is used for many kinds of inference in addition to diagnosis, including probabilistic inference and constraint satisfaction. The tree we generate is a directed tree with a fixed root, and the nodes of the tree are generated based on the clan property; a clique-tree is undirected (with an arbitrary root), and the nodes of the tree are generated based on the family property. One can think of the D-tree as a directed variant of a clique-tree, which is optimized for diagnostic inference. In addition, our approach uses the ordering of the D-tree to require message-passing in a single direction only; in contrast, message propagation in clique trees is bi-directional.

Deb et al. (Deb *et al.*, 1998) describe an implemented approach (based on the TEAMS-RT platform) for performing decentralized diagnosis. This approach shares several similarities to our own, such as the use of local and global specifications for decentralized modules. However, they make a strong assumption that we believe does not hold true in practice: they assume that the outputs of each subsystem are observable. If we make that assumption in our approach, then it is guaranteed that local diagnoses will always equal global diagnoses.

Our work also bears some relation to papers describing distributed solutions to Constraint Satisfaction Problems (CSPs) (Yokoo *et al.*, 1998; Hirayama and Yokoo, 2000). As with the work on distributed logical inference (Amir and McIlraith, 2000), the task of distributed CSPs is finding a satisfying assignment to the variables, when constraints are distributed in a collection of subsets of constraints. Hence the underlying tasks of distributed diagnosis and CSP satisfiability are different. One issue in this work that is similar to diagnostic reasoning is the recording of minimal sets of unsatisfiable clauses as nogoods (Hirayama and Yokoo, 2000). The computation of nogoods is a key step to computing diagnoses (de Kleer and Williams, 1987).

There have been several proposals for using the ATMS (de Kleer, 1986) in a distributed manner, e.g., (Dragoni, 1993; Mason and Johnson, 1989; Malheiro and Oliveira, 2000). Our approach differs from this work in that our approach uses system topology explicitly, whereas these other approaches do not make as extensive a use of topology.

The compilation approach proposed in this article bears some relation to prior work.[9] (Simon and del Val, 2001) presents an empirical comparison of centralized compilation techniques as applied to several areas, of which diagnosis is one. Our future work includes examining the applicability of these compilation techniques within our distributed framework. Compilation is also examined in (McIlraith and Amir, 2001).

There has been some prior work on distributed model-based diagnosis. For example, the approach in (Frohlich *et al.*, 1997) assumes that the diagnosis computed by each distributed agent is globally correct, and examine the case where agents must cooperate to diagnose components whose status is unknown. Our approach makes the more realistic assumption that diagnoses are not necessarily globally sound, and derives a very different global synthesis algorithm.

7 SUMMARY AND CONCLUSIONS

This document has described a mechanism for computing distributed diagnoses using system topology and observability properties. This algorithm takes as input minimal diagnoses computed within distributed com-

[9] A review of compilation can be found in (Cadoli and Donini, 1997).

ponents, and uses system topology to integrate these diagnoses into a globally sound and minimal system diagnosis. We are in the process of applying this approach to two real-world domains, that of In-Flight Entertainment and diagnosis of HVAC systems.

The approach presented here provides a mechanism for designing systems with predictable distributed diagnostics properties. A given decomposition graph can be rated according to its diagnosability and efficiency. Additionally, given a system description, we can apply D-tree decomposition algorithms to the system DAG to assist in identifying small-treewidth decompositions, if any exist. Further, if a system has no small treewidth decomposition, one can then recommend system re-design to help achieve an efficient distributed diagnosis solution.

References

E. Amir and S. McIlraith. Paritition-based logical reasoning. In *Proc. KR '2000*, pages 389–400. Morgan Kaufmann, 2000.

S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Meth.*, 8:277–284, 1987.

Hans L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

H. Bodlander. Treewidth: Algorithmic techniques and results. In *Proceedings 22nd International Symposium on Mathematical Foundations of Computer Science, MFCS'97*, volume 1295 of Lecture Notes in Computer Science, pages 29–36. Springer-Verlag, 1997.

Marco Cadoli and Francesco M. Donini. A survey on knowledge compilation. *AI Communications*, 10(3-4):137–150, 1997.

A. Darwiche and G. Provan. Exploiting system structure in model-based diagnosis of discrete-event systems. In *Proc. 7th Intl. Workshop on Principles of Diagnosis*, pages 95–105, 1996.

Adnan Darwiche. Model-based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research*, 8:165–222, 1998.

J. de Kleer and B. Williams. Diagnosing Multiple Faults. *Artificial Intelligence*, 32:97–130, 1987.

J. de Kleer. An Assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.

S. Deb, A. Mathur, P. Willet, and K. Pattipati. Decentralized real-time monitoring and diagnosis. In *Proc. IEEE SMC Conference*, pages 2998–300, San Diego, CA, 1998.

A. Dragoni. Distributed belief revision versus distributed truth maintenance: preliminary report. In *Atti del 3zo Incontro del Gruppo AI*IA di Interesse Speciale su Inteligenza Artificiale Distribuita*, pages 64–73, Rome, Italy, 1993.

O. Dressler and Peter Struss. The consistency-based approach to the automated diagnosis of devices. In Gerhard Brewka, editor, *Principles of Knowledge Representation*, pages 267–311. CSLI Publications, Stanford, CA, USA, 1996.

Yousri El Fattah and Rina Dechter. Diagnosing tree-decomposable circuits. In *IJCAI*, pages 1742–1749, 1995.

Peter Frohlich, Iara de Almeida Mora, Wolfgang Nejdl, and Michael Schroeder. Diagnostic agents for distributed systems. In *ModelAge Workshop*, pages 173–186, 1997.

K. Hirayama and M. Yokoo. The effect of nogood learning in distributed constraint satisfaction. In *Proceedings of the 20th IEEE International Conf. on Distributed Computing Systems*, pages 169–177, 2000.

T. Johnson, N. Robertson, P. Seymour, and R. Thomas. Directed tree-width. *to appear in J. Combin. Theory Ser. B.*, 2002.

Benedita Malheiro and Eugenio Oliveira. Solving conflicting beliefs with a distributed belief revision approach. In *IBERAMIA-SBIA*, pages 146–155, 2000.

Cindy L. Mason and Rowland R. Johnson. DATMS: A framework for distributed assumption based reasoning. In Les Gasser and Michael N. Huhns, editors, *Distributed Artificial Intelligence*, volume 2, pages 293–317. Pitman, 1989.

Sheila A. McIlraith and Eyal Amir. Theorem proving with structured theories. In *Proc. IJCAI*, pages 624–634. Morgan Kaufmann, 2001.

G. Provan. On the diagnosability of distributed discrete-event systems. In *Proc. 2002 American Control Conf.*, Anchorage, AK, May 8-10 2002.

R. Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32:57–96, 1987.

N. Robertson and P. Seymour. Graph minors. ii. algorithmic aspects of treewidth. *J. Algorithms*, 7:309–322, 1986.

Laurent Simon and Alvaro del Val. Efficient consequence finding. In *IJCAI*, pages 359–370, 2001.

Markus Stumptner and Franz Wotawa. Diagnosing tree-structured systems. *Artificial Intelligence*, 127(1):1–29, 2001.

Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *Knowledge and Data Engineering*, 10(5):673–685, 1998.

A PROOFS OF THEOREMS

This section presents the proofs for the theorems contained in the article.

Theorem 1 *If we have a system description Φ consisting of two component system descriptions Φ_1 and Φ_2, and a system observation θ, if the variables shared by Φ_1 and Φ_2 all appear in θ, then*

$$D^{\Phi}(\theta) \equiv D^{\Phi_1}(\theta_1) \wedge D^{\Phi_2}(\theta_2).$$

Proof: The original statement of this result was in terms of a consequence, i.e.,

$$Cons^{\Phi}(\theta) \equiv Cons^{\Phi_1}(\theta_1) \wedge Cons^{\Phi_2}(\theta_2),$$

where a consequence *Cons* is defined as

Definition Given a system observation θ and system description SD, the consequence $Cons_{\mathcal{A}}^{\Sigma}(\theta)$ is a sentence satisfying the following properties:

1. $Cons_{\mathcal{A}}^{\Sigma}(\theta)$ is an $\mathcal{A}$-sentence;
2. $\Sigma \cup \{\theta\} \models Cons_{\mathcal{A}}^{\Sigma}(\theta)$;
3. For any $\mathcal{A}$-sentence β, $\Sigma \cup \{\theta\} \models \beta$ only if $\Sigma \cup \{\theta\} \models Cons_{\mathcal{A}}^{\Sigma}(\theta)$.

We have restated the theorem replacing consequence with diagnosis. So to prove this result, we just need to show that a diagnosis is a specialized instance of a consequence. By using the third property of consequence, we can write

For any diagnosis ($\mathcal{A}$-sentence) D, $\Sigma \cup \{\theta\} \models D$ only if $\Sigma \cup \{\theta\} \models Cons_{\mathcal{A}}^{\Sigma}(\theta)$. Based on this property, the theorem holds. □

To facilitate the proof of these results, we introduce some notation. We assume, WLOG, that the vertices of $\mathcal{G}_{\mathcal{X}}$ are numbered $V_1, ..., V_n$, where V_1 is the root, and all other nodes are ordered based on a breadth-first expansion from the root. We denote the children of V_i as $\chi(V_i)$. The leaves of $\mathcal{G}_{\mathcal{X}}$ are denoted by $\lambda = \{V_i | \chi(V_i) = \emptyset\}$. In an analogous fashion, we order the clans of a clan graph such that the root is Y_1, and all other nodes are ordered based on a breadth-first expansion from the root. Recall that the system description for V_i is Φ_i, and for the clan $Y(V_i)$, i.e. $V_i \cup \chi(V_i)$, is Φ_i; further, the observable instantiation for clan $Y(N_i)$ is ${V_{obs}}^{\Phi_i}$. We know that ${V_{obs}}^{\Phi_i} = \bigcap_j {V_{obs}}^{\Phi_j} \quad \forall V_j \in Y(V_j)$.

Lemma 1 *If a sound, complete and minimal global diagnosis exists, then no clan diagnosis will have a contradictory diagnosis.*

Proof: Given clan Y_i with observation ${V_{obs}}^{\Phi_i}$ and system description Φ_i, if

$$\Phi_i \cup {V_{obs}}^{\Phi_i} \models \perp, \text{ then}$$

$$\Phi_i \cup {V_{obs}}^{\Phi_i} \cup \alpha \models \perp, \text{ for any sentence } \alpha.$$

Hence, it is not possible to assume that there is a contradiction in a clan yet no contradiction in the complete system, since $\Phi_i \subseteq \Phi$. □

Theorem 2 *Given a tree-structured decomposition graph $\mathcal{G}_{\mathcal{X}}$ and local component diagnoses, diagnostics synthesis will compute a sound and globally consistent set of fault mode assignments for components $\mathcal{X} \in \mathcal{G}_{\mathcal{X}}$ within $O(|\mathcal{Y}|)$ message-passing steps, where $\mathcal{G}_{\mathcal{Y}}$ is the clan graph generated from $\mathcal{G}_{\mathcal{X}}$.*

Proof: We can break this result up into two parts, first the soundness and global consistency, and second the time-complexity.

1. Soundness and global consistency

From Theorem 1, we know that $D^{\Phi}(\theta) \equiv \bigwedge_i D^{\Phi_i}(\theta_i)$. Given a tree, each clan of the tree defines a collection of component system descriptions that must satisfy this theorem. In other words, given a node V_i in the tree, the only variables with which V_i shares variables are its children $\chi(V_i)$ and its parent $\pi(V_i)$. Further, we know that $\pi(V_i)$ is independent of $\chi(V_i)$ given V_i, using a well-known property of (directed) graphs. Hence, once V_i has a complete, sound and minimal (CSM) diagnosis, then the diagnosis for $\pi(V_i)$ can be synthesized independent of $\chi(V_i)$.

We now prove this result inductively for all nodes in the decomposition graph $\mathcal{G}_{\mathcal{X}}$, starting at the leaves of $\mathcal{G}_{\mathcal{X}}$. We assume at the outset that each local diagnosis is complete, sound and minimal (CSM): $D(\Phi_i)$ is CSM, $\forall V_i$. Starting at the leaf clans corresponding to $\mathcal{G}_{\mathcal{X}}$, we synthesize the diagnoses for all leaf nodes λ of $\mathcal{G}_{\mathcal{X}}$, and for the parents of these leaf nodes, $\pi(\lambda)$. We know by Lemma 1 that no clan can ever produce an unsound diagnosis; the issue here is guaranteeing minimality. At the next step, taking any synthesized node $V_j \in \pi(\lambda)$, we can now synthesize its parent in $\mathcal{G}_{\mathcal{X}}$ using the clan $Y(\pi(V_j))$. We know that this synthesis will create a CSM diagnosis for this clan since $\pi(V_j)$ is independent of the values of the nodes in λ, and all the children of V_j have been synthesized through the first synthesis step. We now proceed inductively on the nodes of $\mathcal{G}_{\mathcal{X}}$ or decreasing order to the root. After synthesizing the root clan, we now know that every clan in the clan graph contains globally CSM diagnoses, and hence each node in $\mathcal{G}_{\mathcal{X}}$ has CSM diagnoses.

Finally we need to show that this process terminates with a globally CSM diagnosis. Assume that we have completed the process, yet there is some vertex $V_i \in \mathcal{G}_{\mathcal{X}}$ that does not have a globally CSM diagnosis. Since we know that every node $V_i \in \mathcal{G}_{\mathcal{X}}$ is contained in at least one clan, and every clan has been synthesized, then every node must have been synthesized, and have a CSM diagnosis at the termination of the process. Hence, we have a contradiction, and this process must terminate with a globally CSM diagnosis.

2. Time-complexity If the diagnosis of a child-node V_j in a clan is modified through the synthesis process, then we must perform diagnostic synthesis on $Y(V_j)$. Assume that we start from the leaves of $\mathcal{G}_{\mathcal{X}}$. Given that the clan graph has $|\mathcal{Y}|$ clans, we need to perform $|\mathcal{Y}|$ synthesis steps to reach the root of the tree.

Hence, diagnostics synthesis will terminate and compute a sound and globally consistent set of fault mode assignments for components $\mathcal{X} \in \mathcal{G}_{\mathcal{X}}$ within $O(|\mathcal{Y}|)$ message-passing steps. □

Eliminating Incoherence from Subjective Estimates of Chance

Randy Batsell
Rice University
Houston TX 77251
batsell@rice.edu

Lyle Brenner
University of Florida
Gainesville FL 32611
lbrenner@ufl.edu

Daniel Osherson
Rice University
Houston TX 77251
osherson@rice.edu

Moshe Y. Vardi
Rice University
Houston TX 77251
vardi@rice.edu

Spyros Tsavachidis
Rice University
Houston TX 77251
spy@rice.edu

Abstract

Human judgment is an essential source of Bayesian probabilities but is plagued by incoherence when complex or conditional events are involved. We consider a method for adjusting estimates of chance over Boolean events so as to render them probabilistically coherent. The method works by searching for a sparse distribution that approximates a target set of judgments. (We show that sparse distributions suffice for this purpose.) The feasibility of our method was tested by randomly generating sets of coherent and incoherent estimates of chance over 30 to 50 variables. Even with 50 variables, good approximations were computed within a few hours. Empirical test was provided by asking people to estimate the chances of events relating to the stock market. The estimates of each participant were incoherent but well approximated by the coherent distribution constructed for him. In addition, the quadratic scores of the reconstructed estimates were reliably superior to the scores of the original estimates. In this sense, our correction method improves the objective accuracy of the judgments that it renders coherent. The judgments of all participants were then pooled to form an "aggregate judge," and a coherent approximation to these 1426 aggregated judgments was computed. The quadratic scores for the resulting (coherent) probabilities were superior to the original estimates and also to the coherent approximations computed for subjects individually. Our method thus offers a new approach to aggregating opinions, an important problem in decision theory.

1 Introduction

Human judgment is a fertile source of information about uncertainty in the environment. Intuitive estimates of chance, however, often conflict with the axioms of probability when logically related events must be evaluated. Estimates that cannot be reconciled with any probability distribution are called *incoherent*. The tendency for human judgment to stray from coherence is well documented (Yates, 1990; Osherson, 1995) and is commonly encountered when eliciting probabilities for the construction of Bayesian networks (van der Gaag et al., 1999). Expert judgment in fields like medicine and physics is not immune to such errors, as revealed in experimental studies (e.g., Tversky & Kahneman, 1983; Viale & Osherson, 2000).[1] The difficulty people experience in maintaining coherence may stem in part from the intrinsic complexity of probability manipulations (Georgakopoulos et al., 1988).

One way to ensure probabilistic consistency is to pair each query to a judge with a precalculated response set containing all coherent possibilities (cf. Druzdzel & van der Gaag, 1995). Such structured elicitation can be tedious, however, since verifying coherence may involve large calculations. The estimates of chance finally obtained, moreover, may depend on the order in which they were generated because later estimates are more highly constrained than earlier ones. As a complement to structured elicitation, it is therefore desirable to possess a method for adjusting estimates of chance off line, rendering them coherent after the judge has left the scene. An obvious constraint on the method is to return coherent estimates that are as close as possible to the original judgments (an exact match

[1] Indeed, it is striking to observe how many well educated people find nothing exceptional about ascribing probability 30% to Man reaching Mars by 2020, 80% probability of a sustained global economic downturn starting in 2010, and 5% probability to the conjunction of these events.

can be achieved only if the judgments are coherent to begin with).

The present paper advances such a method. It presents an algorithm that accepts a corpus of probabilistic judgments (typically incoherent) and returns a revised corpus that is consistent with some joint distribution over all the events in play. The goal of the algorithm is to choose probabilities that are close to the original estimates. A successful algorithm of this kind can be used not only to amend inconsistent estimates of chance from a single judge. It can also suggest a consensus view among a group of experts whose judgments cannot be coherently pooled. The pooled judgments may be submitted to the algorithm in view of producing a single corpus of probabilistic estimates that minimally distorts the views of the individual judges. (Results on "aggregating" opinion in this way are reported below in Section 4.1.5.)

The principal challenge for the algorithm we seek is manipulating large probability distributions. Recall that a distribution over n Boolean variables assigns probabilities to 2^n elementary states (called "truth assignments" in what follows). Since practical applications can involve scores of variables, it is necessary to represent underlying distributions in compact form. For this purpose we rely on a simple data structure called a *probability array* (or *array*), described below. It will be shown that small arrays suffice to optimally approximate virtually any set of incoherent estimates of chance. To search for an array that approximates a target set of incoherent judgments, we rely on simulated annealing (van Laarhoven, 1988). As reported below, searching through the class of probability arrays via simulated annealing allowed us to efficiently approximate target sets of simulated judgments over 30 to 50 variables. The same method was applied to sets of 46 judgments over 10 variables in four empirical studies involving 134 human judges. Close approximations were again achieved efficiently. In the empirical studies, the *quadratic scores* of the reconstructed estimates were reliably superior to the scores of the original estimates. (The quadratic score, defined below, is a familiar measure of the objective accuracy of an estimate of chance.) Even better quadratic scores resulted from pooling the judgments of all participants in a given study into an "aggregate judge," and then applying our algorithm to achieve a coherent approximation. Thus, in both the individual and aggregate sense, our correction method improves the objective accuracy of the judgments that it renders coherent. Thus, the approach advocated here is both feasible and shown to improve the predictive power of judges, as well as to enable aggregation of the judges' estimates (a well known problem is decision theory, cf. Adler & Ziglio, 1996).

In place of probability arrays, we have also experimented with *algebraic decision diagrams* (ADDs), which have proven useful in other problems involving numerous variables (Bahar et al., 1997). Similarly, we have explored genetic algorithms as an alternative to simulated annealing. We compared the performance of the two approaches (probability arrays with simulated annealing and ADDs with genetic algorithms). The two approaches construct coherent approximations of similar quality, but differ markedly in their running times.

To proceed, we first specify the optimization problem posed by off line reconstruction of probability estimates. Then we give details about our optimization method involving arrays and simulated annealing. Experimental results are described for this method. We then describe the use of algebraic decision diagrams in conjunction with genetic algorithms.

2 Off line reconstruction as optimization

Finding a coherent approximation to incoherent estimates of chance amounts to solving an optimization problem. To state the matter formally, let $v_1 \cdots v_n$ be Boolean variables, representing the occurrence or non-occurrence of n logically independent events. Syntactically, the variables give rise to an infinity of formulas built up in the usual way from sentential connectives like $\neg, \wedge, \vee$. The formulas serve to describe absolute events whereas pairs $(\varphi : \psi)$ of formulas describe conditional events. Semantically, we have 2^n mappings (called *truth assignments*) from $\{v_1 \cdots v_n\}$ to $\{true, false\}$. A truth assignment α *satisfies* a formula φ if φ evaluates to *true* under α via standard propositional logic semantics. A *(probability) distribution* over $v_1 \cdots v_n$ is a mapping of the 2^n truth assignments into nonnegative numbers that sum to unity. Distribution Pr is extended to formulas φ via: $\Pr(\varphi) = \Sigma\{\ \Pr(\alpha) : \alpha \text{ satisfies } \varphi\ \}$. It is extended to pairs $(\varphi : \psi)$ of formulas via:

$$\Pr(\varphi : \psi) = \frac{\Sigma\{\ \Pr(\alpha) : \alpha \text{ satisfies both } \varphi \text{ and } \psi\ \}}{\Sigma\{\ \Pr(\alpha) : \alpha \text{ satisfies } \psi\ \}}$$

provided that $\Pr(\psi) > 0$. If $\Pr(\psi) = 0$ then $\Pr(\varphi : \psi)$ is undefined.

Consider a judge who is estimating the probabilities of some events and conditional events over $v_1 \cdots v_n$. We write $Prob(\varphi) = x$ to indicate the judgment that the probability of φ is x, and $Prob(\varphi : \psi) = y$ for the

judgment that the conditional probability of φ assuming ψ is y. *Prob* is thus a finite function from formulas and pairs of formulas to numbers. If it coincides on its domain with some distribution Pr over $v_1 \cdots v_n$ then *Prob* is called *coherent*, otherwise *incoherent*. In the typical case, *Prob* is incoherent, and we seek to reconstruct it via a close coherent distribution. The resulting COHERENT APPROXIMATION PROBLEM is:

(1) *Let Prob map formulas $\varphi_1 \cdots \varphi_k$, and pairs of formulas $(\chi_1, \psi_1) \cdots (\chi_j, \psi_j)$, into $[0, 1]$. Find a map Prob* with the same domain as Prob such that Prob* is coherent, while minimizing*

$$\sum_{i \leq k} \mid Prob(\varphi_i) - Prob^*(\varphi_i) \mid +$$

$$\sum_{i \leq j} \mid Prob(\chi_i : \psi_i) - Prob^*(\chi_i : \psi_i) \mid .$$

Squared deviation from *Prob* could be substituted for the absolute deviation appearing in (1). In either case, it is easy to see how to convert an efficient solution to the Coherent Approximation Problem — or to its squared deviation version — into a method for testing the satisfiability of Boolean formulas, implying that the optimization problem is NP-hard.[2] We prefer absolute to squared deviation because it allows the use of linear programming in special cases (see below).

Note that distinct approximations *Prob** can yield the same distance to *Prob* yet assign different probabilities to the target events $\varphi_1 \cdots \varphi_k$, and $(\chi_1, \psi_1) \cdots (\chi_j, \psi_j)$. For example, the incoherent judgments $Prob(p) = .3$, $Prob(\neg p) = .6$ are equally well reconstructed as $Prob^*(p) = .4$, $Prob^*(\neg p) = .6$ or as $Prob^*(p) = .3$, $Prob^*(\neg p) = .7$. Such multiplicity invites additional desiderata in the formulation of the Coherent Approximation Problem, for example, maximizing entropy. For simplicity in the current investigation, only deviation from *Prob* appears in our formulation (1). Absolute and conditional events, moreover, are given equal weight in calculating deviation from *Prob*, and only point estimates of probability are considered. Obviously, such matters can be settled differently within specific applications.

A special case of the Coherent Approximation Problem can be solved by linear programming (LP). If only the probabilities of absolute events are estimated, then LP can be used as follows to calculate the closest possible approximation. For each truth assignment α we have a variable x_α, and for each (absolute) event e we have a variable k_e. For each estimate p associated with an event e, we have two constraints. One has the form $k_e + \Sigma x_\alpha \geq p$, where the summation ranges over the truth assignments α that satisfy e. The other one has the form $-k_e + \Sigma x_\alpha \leq p$, with the same summation. The two constraints thus amount to $\mid p - \Sigma x_\alpha \mid \leq k_e$. A final constraint sets the sum of all the x_α's to unity. Note that this LP formulation of the Coherent Approximation Problem has an exponential number of variables, but the number of constraints is linear in the number of probability estimates. The objective is to minimize the sum of the k_e's. The resulting distribution (given by the x_α's) minimizes this sum and can be shown to yield a best approximation in the sense of (1).[3]

LP provides a useful benchmark of success in simple settings but it does not embody a general solution to the Coherent Approximation Problem. For one thing, LP is impractical for more than 30 variables (requiring a constraint matrix with more than a billion columns). More importantly, it cannot be applied to judgments involving ratios of probabilities, notably, when estimates are given for conditional events. We now describe a more general approach to the Coherent Approximation Problem. and thus a more general approach to off line reconstruction of incoherent judgments.

3 Optimization method

To confront the Coherent Approximation Problem, our strategy is to represent candidate distributions using probability arrays, and to search through them via simulated annealing. We consider arrays and annealing in turn.

A *probability array* of size (n, m) (for $n, m > 0$) is a set of m vectors each of form $\alpha_1^i \cdots \alpha_n^i, \beta^i$ $(1 \leq i \leq m)$, where $\alpha_k^i \in \{1, 0, *\}$, $\beta^i \in [0, 1]$, and $\sum_{i=1}^m \beta^i = 1$. Letting the vectors be columns, one array of size $(3, 4)$ may be pictured as in (2)a, below.

[2] In addition to the problem of intractability is the possibility that there may not even be a minimum distance between *Prob* and a coherent approximation *Prob** to it. The incoherent judgments $Prob(p : q) = .5$, $Prob(q) = 0$, for example, can be approximated by setting $Prob^*(p : q) = .5$ and $Prob^*(q)$ arbitrarily close to 0, but not 0 itself.

[3] Neither the distribution nor the optimizing k_e's are unique. Many choices of the k_e may produce minimum deviation from the target estimates.

(2)

(a)

1	0	*	0
0	*	*	0
1	0	1	0
.2	.1	.4	.3

(b)

1	1	1	1
0	0	0	0
1	1	1	1
.2	.1	.4	.3

(c)

*	*	*	*
*	*	*	*
*	*	*	*
.2	.1	.4	.3

The first three rows of (2)a correspond to three (Boolean) variables v_1, v_2, v_3, respectively. The first column represents the truth-assignment v_1 = true, v_2 = false, v_3 = true, and assigns it probability .2. The second column represents two truth-assignments, namely, v_1 = false, v_2 = true, v_3 = false, and v_1 = false, v_2 = false, v_3 = false. In other words, the symbol * functions as "don't care." The two truth-assignments represented by the second column of (2)a share the value .1, each receiving .05. The other two columns are interpreted similarly. (Four truth-assignments are represented in the third column, and are each assigned .4/4. The sole truth-assignement represented by the fourth column is assigned .3.) Notice that the same truth-assignment may appear in more than one column. The probability of such a truth-assignment is the sum of the values it receives in each column where it appears. For example, in (2)a, the truth-assignment v_1 = true, v_2 = false, v_3 = true is represented in columns 1 and 3. Its probability according to (2)a is $.2 + (\frac{1}{4} \times .4) = .3$. Some truth-assignments may not appear anywhere in the array, and are therefore assigned probability zero. For example, in (2)a, the probability of v_1 = true, v_2 = true, v_3 = false is 0. As an aid to intuition, the extreme arrays (2)b,c may be helpful. The first represents a highly sparse distribution, placing all probability on one truth-assignment. The second represents the uniform distribution.

It should be clear that every array of size (n, m) represents a distribution over n variables. It should also be clear that:

(3) FACT: *Every distribution over n variables that assigns positive probability to no more than m truth-assignments is represented by some array of size (n, m).*

Let us now note that following fact (exploited in a similar way by Fagin et al., 1990).

(4) FACT: *Let formulas $\varphi_1 \cdots \varphi_k$, and pairs $(\chi_1, \psi_1) \cdots (\chi_j, \psi_j)$ of formulas be given. For every distribution* Pr *there is a distribution* Pr' *such that:*

(a) Pr' *assigns positive probability to at most $k+j+1$ truth assignments;*

(b) $\Pr'(\varphi_i) = \Pr(\varphi)$ *for every $1 \leq i \leq k$;*

(c) $\Pr'(\chi_i : \psi_i) = \Pr(\chi_i : \psi_i)$ *for every $1 \leq i \leq j$ with $\Pr(\chi_i : \psi_i)$ defined;*

Proof of Fact (4): Let k formulas and j pairs of formulas be given as in (4). Suppose that all the formulas are written over the same set of n variables. Let Pr be a probability distribution for the formulas and pairs of formulas. Hence, Pr maps the 2^n truth assignments $\tau_1, \tau_2, ...\tau_{2^n}$ into non-negative real numbers $x_1, x_2, ..., x_{2^n}$ such that $\sum_{i=1}^{2^n} x_i = 1$. Let

$$\begin{aligned} S_i &= \{\ell : \tau_\ell \text{ satisfies } \varphi_i\} \quad \text{for } i \leq k, \\ Y_i &= \{\ell : \tau_\ell \text{ satisfies } \chi_i\} \quad \text{for } i \leq j, \\ Z_i &= \{\ell : \tau_\ell \text{ satisfies } \psi_i\} \quad \text{for } i \leq j. \end{aligned}$$

It then follows that:

$$\Pr(\varphi_i) = \sum_{\ell \in S_i} x_\ell \quad \text{for } i \leq k,$$

$$\Pr(\chi_i : \psi_i) = \frac{\sum_{\ell \in Y_i \cap Z_i} x_\ell}{\sum_{\ell \in Z_i} x_\ell} \quad \text{for } i \leq j.$$

The latter equalities imply that the following equations have a nonnegative real solution.

$$\sum_{\ell \in S_i} x_\ell = \Pr(\varphi_i), \quad \text{for } i \leq k,$$

$$\sum_{\ell \in Y_i \cap Z_i} x_\ell - \Pr(\chi_i : \psi_i) \cdot \sum_{\ell \in Z_i} x_\ell = 0, \quad \text{for } i \leq j,$$

$$\sum_{i=1}^{2^n} x_i = 1.$$

It is well known that if a system of m linear equations in p unknowns has a nonnegative solution then it has a solution with at most m nonnegative values (see, e.g., Chvátal, 1983, Thm 9.3). It thus follows immediately that there is another solution to the last equations with at most $k+j+1$ non-negative entries. The desired sparse probability distribution Pr' consists of these values. ■

From (4) it follows that our search for an approximating distribution (in the sense of (1)) can be limited to sparse distributions. Hence (3) implies that the goal of the search can be restricted to the class of probability arrays. Specifically, if the judge has estimated probabilities for m events and conditional events over n variables, then an optimal coherent approximation is represented by some array of size $(n, m+1)$. The

optimal array need not include *, but the presence of *'s allows optimal approximation in some cases by even smaller arrays. Furthermore, the presence of *'s allows the search to go through intermediate points that cannot be represented by a *-free array of size $(n, m+1)$. We come back to the issue of *'s later.

We now turn to the search algorithm. Simulated annealing is described in many places (e.g., van Laarhoven, 1988). It suffices here to explain the selection of initial points, and also how neighbors to a given point were constructed. As initial points for a problem involving m events over n variables, we randomly generated a set of probability arrays of size $(n, \hat{m})$, where $\hat{m}$ was determined empirically in light of our experience finding good approximations (specific choices of $\hat{m}$ are reported below). Each cell of an array had probability $\hat{p}$ of being assigned *, and otherwise equal probability of being assigned either 1 or 0 ($\hat{p}$ was also determined empirically). The last row of each column (the probabilities) were uniformly randomly chosen from $[0, 1]$ (using double-precision floating-point numbers), then renormalized to sum to 1.0. For creating neighbors, the algorithm relied on the following routine.

> Routine for creating neighbors to an array $\mathcal{A}$: for every column in $\mathcal{A}$, with 10% probability (a) randomly choose one of the top n entries and replace its entry with a randomly chosen member of $\{0, 1, *\}$, and (b) multiply the last entry (namely, the probability) by a random choice between .9 or 1.1, then renormalize across columns (so that they sum again to 1.0).

The initial and final temperatures, as well as the cooling schedule for the algorithm conformed to the recommendations in van Laarhoven (1988). The number of temperature steps in the algorithm was determined empirically, as was the number of iterations at a given step. When we employed multiple starting points, they were managed via the procedure known as "go with the winners" (Aldous & Vazirani, 1994).

4 Tests of the method

We first tested our method by soliciting estimates of chance for logically simple and complex events over sets of 10 variables. To assess its scalability, we then tested the method against simulated judgments over 30+ variables. These tests are described in turn. All experiments were run on a Pentium III PC running at 533 MHz within the Java Virtual Machine for Windows '98.

4.1 Empirical studies

4.1.1 Data collected

Consider the following event: "The change in value of United Airlines stocks in the third quarter of 2000 will be more favorable than the change in value of the S&P 500 composite." Nine other events of the same form were constructed involving the companies Continental, American, Southwest, Exxon, Chevron, Texaco, British Petroleum, Enron, and Schlumberger. The above event was abbreviated to "United Airlines outperforms the S&P 500," and similarly for the other companies.

In the summer of 2000, 26 MBA students at the Jones School of Management (Rice University) along with 5 professional stock traders estimated the probabilities of the ten events plus the probability of 36 complex events built from them.[4] The 36 complex events included an individually randomized selection of 6 events from each of the following categories.

(a) CONDITIONAL EVENTS like "Exxon outperforms the S&P 500 *assuming that* Chevron outperforms the S&P 500."

(b) CONDITIONAL EVENTS WITH NEGATION like "Exxon outperforms the S&P 500 *assuming that* Chevron does not outperform the S&P 500."

(c) CONJUNCTIVE EVENTS like "Exxon outperforms the S&P 500 *and* Chevron outperforms the S&P 500."

(d) CONJUNCTIVE EVENTS WITH NEGATION like "Exxon outperforms the S&P 500 *and* Chevron does not outperform the S&P 500."

(e) DISJUNCTIVE EVENTS like "Exxon outperforms the S&P 500 *or* Chevron outperforms the S&P 500."

(f) DISJUNCTIVE EVENTS WITH NEGATION like "Exxon outperforms the S&P 500 *or* Chevron does not outperform the S&P 500."[5]

The six events in each category were chosen individually randomly for each participant from the 45 nontrivial possibilities (thus, different events were chosen

[4] The data from MBA students and traders were not distinguishable so they are treated together.

[5] The inclusive meaning of *or* was explained to all participants prior to collecting their estimates. Other instructions clarified the conditional reading of the expression *assuming that*, as well as the exact financial significance of outperforming the S&P 500 index.

for different participants). The participants first made probability estimates for the 10 variables, then for the 36 complex events in individualized random order under the constraint that the six events in a given category appear as a block. The 46 queries were administered via a web interface at the participant's convenience and required about half an hour to collect.

4.1.2 Incoherence

Not one of the 31 participants offered a coherent set of estimates. For example, they averaged 3.0 violations (out of 6 possible) of the following constraint on the probabilities of conjunctions of form $p \wedge q$.

$$(5)\ \Pr(p)+\Pr(q)-1 \leq \Pr(p \wedge q) \leq \min\{\Pr(p), \Pr(q)\}.$$

Similarly, they averaged 3.23 violations (out of 6) of the following constraint on disjunctions of form $p \vee q$.

$$(6)\ \max\{\Pr(p), \Pr(q)\} \leq \Pr(p \vee q) \leq \Pr(p) + \Pr(q).$$

Constraints corresponding to conditional events yielded comparable rates of violation.

4.1.3 Approximation via simulated annealing over arrays

We applied our algorithm to each participant separately, represented by his/her dataset of 46 judgments. There were ten starting points, 150 temperature steps, and 25 iterations per step. Arrays were of size $(10, 50)$. The probability of $*$ was set to .2. With these parameters, the algorithm required less than 20 seconds per dataset. We experimented with different number of columns (25, 50, and 100) and different probabilities of $*$ (0, 0.2, and 0.4). The running time scales roughly linearly with the number of columns, but was not affected by the probability of $*$. Neither the number of columns nor the probability of $*$ had a measureable impact on the quality of the approximation, indicating the robustness of the technique.

For each participant we calculated the *mean absolute deviation* (MAD) between her 46 probability estimates and those offered by the best approximating array discovered for her. Across the 31 participants, the average MAD was .095 (S.D. = .043). Thus, to render coherent our judges' estimates of chance, we modified them by adding or subtracting roughly .1, on the average.

4.1.4 Comparison to linear programming

Perfect approximation (MAD of 0) by a probability distribution is ruled out because of the incoherence that characterizes the estimates. For just the 34 estimates of absolute events the closest possible approximation can be calculated using linear programming (LP) as discussed in Section 2 above. We applied LP to each of the 31 participants individually. The average MAD obtained was .081 (S.D. = .049). For comparison purposes, we again applied our simulated annealing algorithm to each participant after deleting the estimates of conditional events. (Both LP and the second application of simulated annealing involved 34 absolute estimates per participant.) The average MAD achieved was .093 (S.D. = .049). Thus, for absolute events, the simulated annealing algorithm came within 15% of producing an optimal approximation.

4.1.5 Impact on accuracy

We desire to improve the estimates of a judge by making them coherent. But if the process robs the judgments of their accuracy, the judge might prefer that we left her estimates in their original state. The accuracy in a judgment can be measured via its quadratic score, defined as follows (see von Winterfeldt & Edwards, 1986, for general discussion of scoring rules).

Suppose that *Prob* represents the estimates of a given judge. Let E be an event in the domain of *Prob*, and let $(G : F)$ be a pair of events in the domain of *Prob*.

- The quadratic score incurred by *Prob* for E is $(1 - Prob(E))^2$ if E is true. It is $Prob(E)^2$ if E is false.
- The quadratic score incurred by *Prob* for the pair $(G : F)$ is $(1 - Prob(G : F))^2$ if both G and F are true. It is $Prob(G : F)^2$ if G is false and F is true. It is not defined if F is false.

The *overall quadratic score* of *Prob* is the average of all the scores incurred by *Prob* for events and pairs of events in its domain. (Pairs of events for which the quadratic score is not defined do not figure in this average.) Notice that the quadratic score is best interpreted as a penalty; low scores signal accurate judgment. If a judge estimated every probability at .5, her overall quadratic score would be .25. While such estimates would be incoherent in our experiment, an overall score below .25 is a convenient benchmark of knowledge about the domain in question.

After the facts were established about the third quarter performance of our 10 stocks, we computed the overall quadratic score for each of the 31 participants separately. The average, overall score was .254 (S.D. = .056). For each participant, we then computed the overall quadratic score associated with the

reconstructed (coherent) estimates due to our algorithm. (The same events and conditional events figure in the scores associated with the original estimates and with their coherent reconstruction.) The average, overall quadratic score for the coherent approximations was .232 (S.D. = .057), which is reliably lower than the original scores ($p < .001$ by correlated t test). The scores for 23 of the 31 participants were lower when computed with the coherent approximation rather than the original estimates. A majority of this size is unlikely to arise by chance ($p < .01$ by a binomial test).

For another assessment of accuracy, define a participant's *discrimination index* to be the average probability assigned to true events minus the average probability assigned to false ones. Conditional events figure in the discrimination index provided that the conditioning events are true. Good judges assign higher probabilities to events that come true, hence have higher discrimination indexes. (See Yates, 1990, Ch. 3 for discussion.) The average discrimination index for the 31 participants was .129 (S.D. = .176) whereas the average discrimination index for the 31 coherent approximations was .180 (S.D. = .156). Once again, this difference is reliable ($p < .001$, by correlated t test). The discrimination index for 26 of the 31 participants improved in the transition from raw to reconstructed estimates.

Overall, our participants seem not to have enjoyed much insight into the stock market. (Thus, their average quadratic score was worse than what could be achieved by responding with .5 to each query.) It is nonetheless clear that our method of coherent reconstruction did not make their judgment worse. Rather, it produced a reliable increase in accuracy.

Next we pooled all the judgments from our 31 participants to form an "aggregate judge" with $31 \times 46 = 1426$ judgments. Applying our algorithm[6] to the aggregate judge resulted in MAD = .203 (after 2 minutes of computation).[7] Three quadratic scores were then compared, namely, (a) the average quadratic score for the original judgments of the individual judges, (b) the average quadratic score for the coherent judgments that result from applying our algorithm to the individual judges, and (c) the quadratic score for the coherent judgments that result from applying our algorithm to the aggregate judge. The respective quadratic scores were .253, .231, and .198. By a correlated t-test, the quadratic score in (c) (that is, for the corrected aggregate judge) was reliably lower than the two others ($p < .001$). For 23 of the 31 participants, the aggregate quadratic score (c) was lower than both the scores associated with original judgments (a) and with individual correction (b). Thus, our set of judges has statistically significant "collective wisdom", even though as individuals their estimates are generally poor.

4.2 Other judgment experiments

We performed three other experiments of identical design. Each involved 10 variables and 36 complex events of forms (a) - (f). One experiment concerned weather prediction and variables like:

> One week from today at noon, it will be at least 58 degrees in Philadelphia.

Five cities and two weather conditions (temperature and precipitation) gave rise to the ten variables. Another experiment required predicting the outcome of a National Basketball Association game between the Houston Rockets and the Phoenix Suns. The ten variables included:

- The Rockets lead at halftime.
- The Rockets have fewer turnovers than the Suns.

The last experiment also involved the NBA, this time a game between the Rockets and the Dallas Mavericks.[8] Let us call the four experiments weather, Suns, Mavericks, and stocks, respectively. There were 38 participants in weather, 29 in Suns, and 36 in Mavericks. All were Rice undergraduates. Except for some overlap between Suns and Mavericks, the sets of participants in the four experiments were disjoint.

We performed the same analyses for weather, Suns, and Mavericks as we did for stocks. The results are now summarized, starting with the incoherence of the participants' estimates. The average numbers of violations of (5) in weather, Suns, and Mavericks were 2.92, 4.21, and 3.64, respectively. For (6), these numbers are 2.16, 3.86, and 2.67. The average MAD between a participant's estimates and those supplied by our coherent approximations was .085 for weather, .115 for Suns, and .094 for Mavericks. Applying LP to just the 34 absolute events yielded coherent approximations with

[6] Again, the results are robust with respect to the number of columns and the probability of $*$; 100 columns suffice, even though we have 1426 judgments.

[7] Higher MAD for the aggregate subject compared to the average MAD for the individual subjects is virtually inevitable since the aggregate involves a proper superset of the judgments of each individual.

[8] Monetary incentives were offered for accurate probabilities in the two basketball experiments.

MADs of .070 for weather, .107 for Suns, and .081 for Mavericks. In comparison, our method applied just to absolute events gave rise to MADs of .083 for weather, .118 for Suns, and .092 for Mavericks. Simulated annealing thus performs within roughly 16% of optimum for absolute events. The average quadratic score for the participants in weather was .232. For Suns and Mavericks the average quadratic scores were .237 and .228, respectively. The quadratic scores for the coherent approximations in weather, Suns, and Mavericks were .201, .203, and .200, respectively. In weather, 30 of the 38 participants improved their quadratic scores in the transition from raw to reconstructed estimates. Improvement occurred for 26 of the 29 participants in Suns, and for 34 of the 36 participants in Mavericks. The average discrimination indexes for the raw estimates in weather, Suns, and Mavericks were .154, .138, and .133, respectively. The average discrimination indexes for the reconstructed estimates in the three experiments were .199, .224, and .216, respectively. Discrimination indexes improved for 34 of the 38 participants in weather, for 25 of the 29 participants in Suns, and for 34 out of 36 participants in Mavericks.

To determine the impact on accuracy of aggregation, for each of weather, Suns, and Mavericks, we pooled all the judgments from the respective participants to form an "aggregate judge." Applying our algorithm to the aggregate judges resulted in respective MADs of .175, .205 and .183. For each study the same three quadratic scores were compiled as before, namely, (a) for the original judgments of the aggregate judge, (b) for the coherent judgments that result from applying our algorithm to the participants individually, and (c) for the coherent judgments that result from applying our algorithm to the aggregate judge. In the case of weather, the respective quadratic scores were .231, .210, and .191, all reliably different by a correlated t-test ($p < .001$). For 30 of the 38 participants, the aggregate quadratic score (c) was lower than the raw quadratic score (a), and (c) was lower than the individually corrected quadratic score (b) for 27 of the 38 participants. These proportions are greater than expected by chance ($p < .01$ by a binomial test). In the case of Suns, the respective quadratic scores were .237, .203, and .166, all reliably different by correlated t-test ($p < .001$). For 28 of the 29 participants, the aggregate quadratic score (c) was lower than the raw quadratic score (a), and (c) was lower than the individually corrected quadratic score (b) for 23 of the 29 participants ($p < .01$ by binomial test). Finally, for Mavericks, the respective quadratic scores were .228, .200, and .172, all reliably different by correlated t-test ($p < .001$). For 30 of the 36 participants, the aggregate quadratic score (c) was lower than the raw quadratic score (a), and (c) was lower than the individually corrected quadratic score (b) for 28 of the 36 participants ($p < .01$ by binomial test).

4.3 Scalability studies

As a test of the computational feasibility of our method in large problems, we created sets of simulated estimates of chance and tried to match them with our algorithm. In some cases the target estimates were constructed to be coherent because the ideal level of accuracy in this case is known (namely, a MAD of zero). To simulate incoherent estimates, we started with coherent probabilities, then perturbed them randomly by adding or subtracting a constant k. (A coin toss determined whether to add or subtract under the restriction that the resulting number remain in the unit interval.) In different tests, k was chosen to be either .1 or .2 (or else zero in the coherent case). Note that the value of k provides an upper bound on the lowest MAD that can be achieved between the original judgments and their coherent approximations.

Nine tests were carried out, involving distributions with 30, 40 and 50 variables. For each distribution, 20,000 truth-assignments were randomly chosen to carry positive probability. To ensure that the distribution was strongly nonuniform (the only challenging case), we proceeded as follows. Truth assignments can be naturally ordered by the binary numbers they encode (relative to a prior ordering of variables). Let α_i ($1 \leq i \leq 2^n$) be one such ordering. For each α_i obtaining positive probability, a number r was uniformly randomly selected in the interval $(0, 1)$, then multiplied by $1 + (.01 \times i)$ and assigned to α_i. Normalization then ensured that the distribution sums to unity. Multiplying by $1 + (.01 \times i)$ in the prenormalization step skews the distribution.

Events were represented by randomly constructed formulas in disjunctive normal form. Specifically, for each event, we randomly chose a number between 1 and 5 to serve as the number of conjunctions, then for each conjunction we randomly chose between 1 and 10 variables with randomly determined polarity to serve as conjuncts. (The variables were drawn from a set of size 30, 40, or 50, depending on the simulation.) Conditional events were constructed in the same way from pairs of events, the second serving as conditioning event.

For each of the 9 tests, we generated 100 events and 100 conditional events. Their probabilities were computed relative to a distribution of the kind described above, then perturbed if incoherence was involved. For events involving n variables, we performed simulated

vars.	30		40		50	
Inc.	MAD	Min	MAD	Min	MAD	Min
0	.0063	132	.0054	154	.0062	140
.1	.0638	135	.0638	140	.0680	144
.2	.1428	85	.1472	104	.1454	215

Table 1: Mean absolute deviation (MAD) and time in minutes (Min) achieved on simulated data generated from sparse distributions with 20,000 positive states. Running times are relative to a Pentium III (533 MHz) processor and the Java Virtual Machine for Windows '98.

annealing over arrays of size $(n, 100)$, with one starting point, 150 temperature steps, 25 iterations per step, and probability of * set to .6. (Again, the algorithm was quite robust with respect to the number of columns. Here, however, the probability of $*$ played an important role; setting it to zero resulted in the algorithm not converging.)

Results are shown in Table 1. To interpret the table, consider the simulation involving coherent judgments and 50 variables. The simulated annealing algorithm produced a coherent approximation within 140 minutes whose mean absolute deviation from the original 200 judgments was .0062. For another example, consider again 50 variables, with probabilities that are rendered incoherent by adding or subtracting .1. Within 144 minutes, the algorithm produced a coherent approximation whose mean absolute deviation from the original 200 judgments was .0680. These results provide evidence for the scalability of our approach to reconstructing incoherent estimates of chance.

5 An alternative optimization technique

Probability distributions may also be compactly represented by *algebraic decision diagrams* (ADDs) instead of probability arrays. Indeed, ADDs have already proven to be a useful data structure in several contexts (Bahar et al., 1997). In this section, we describe how ADDs can be used to find a coherent approximation to incoherent judgments.

5.1 Algebraic decision diagrams

An ADD is a generalization of a *reduced, ordered, binary decision diagram* or ROBDD (Bryant, 1986). The latter structure is a DAG with terminal nodes labeled either 0 or 1, and non-terminal nodes with out-degree

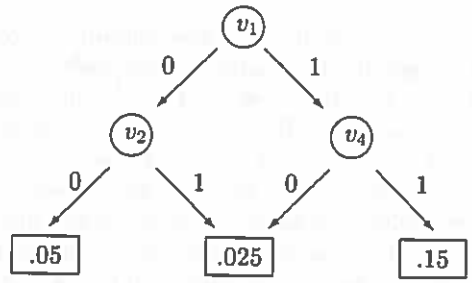


Figure 1: An ADD for a distribution over the variables v_1, v_2, v_3, v_4. According to the distribution, $\Pr(\bar{v}_1\bar{v}_2\bar{v}_3\bar{v}_4) = .05$, $\Pr(\bar{v}_1\bar{v}_2v_3\bar{v}_4) = .05$, $\Pr(v_1\bar{v}_2\bar{v}_3v_4) = .15$, $\Pr(v_1\bar{v}_2\bar{v}_3\bar{v}_4) = .025$, etc.

two labeled by variables. One outgoing edge from a given variable has label 0, the other 1. The DAG is ordered in the sense that all its paths respect a fixed linear ordering of the variables. It is reduced in the sense that (a) two nodes with the same label and having the same 0- and 1-successors (if any) are identified, and (b) there is no nonterminal node whose 0- and 1-successors are identical. An ADD is like an ROBDD except that its terminals can assume any real value. See Figure 1. An ADD maps each truth assignment to one of its terminal nodes in the obvious way. For example, the ADD of Figure 1 maps the uniform assignment of *true* to .15 via its rightmost edges (v_2 and v_3 are treated as "don't cares" on this path). Given an ordering of the variables, a given mapping of truth assignments to numbers is represented by a unique ADD. See Bahar et al. (1997) for the theory of ADDs, and applications. (To avoid misunderstanding, we stress that Figure 1 does not represent a Bayesian network.)

In the present context, we restrict attention to ADDs that yield probability distributions, namely, whose terminals are nonnegative and such that $\Sigma\,\text{terminal}(\alpha) = 1$ where the sum is over all truth assignments α, and $\text{terminal}(\alpha)$ is the value of the terminal node on the path that represents α. In principle, every probability distribution represented by an ADD can also be represented by a probablity array, but the ADD representation might be much more compact. To manipulate ADDs, we rely on a subset of operations defined in the software package CUDD from Colorado University, primarily, on inner product (via matrix multiplication) and on variable reordering.[9] The latter operation seeks to minimize the storage requirements of a set of ADDs through choice of a single ordering of variables for them all. (This operation is heuristic in character.) Inner product is used as follows to

[9]The package may be downloaded via: `http://vlsi.colorado.edu/~fabio/CUDD/cuddIntro.html`

calculate the probability of a given formula φ according to a probability distribution represented by a given ADD A. We first represent φ as a Boolean function by means of an ROBDD B (i.e, an ADD with terminals limited to 0 and 1). A and B can be conceived as vectors over the same set of truth assignments. The desired probability is therefore obtained via their inner product. The time-complexity of this operation is linear in the product of the number of nodes in A and B.

5.2 Genetic algorithms applied to ADDs

To find a coherent approximation to input judgments, we search through the space of probability distributions represented by ADDs, using genetic algorithms.[10] We now provide details about the genetic algorithm used to produce the results presented in the sequel. (For general properties of genetic algorithms, see Mitchell, 1996.) We describe (a) the starting population, (b) the processes of crossover and mutation, and (c) the construction of the next generation. The numerical parameters cited below (e.g., for mutation and crossover) were determined by trial-and-error. The same parameters are used in all the experiments reported later.

5.2.1 Initial population

To form the initial population, a set of ADDs is generated randomly. A given ADD is created as follows. Suppose there are n variables, $v_1 \cdots v_n$. First, a set of m ternary sequences of length n is randomly generated (we chose $m = 50$). Each coordinate of a given sequence holds either *true*, *false*, or *don't care*. For each sequence, a randomly chosen probability p is chosen. By identifying coordinate i of a given sequence with v_i, each ternary sequence represents a set of truth assignments, all with probability p. In essence, such a sequence is a single column of a probability array. An ADD is created for each such set of truth assignments (one ADD for each ternary sequence). The different ADDs corresponding to the different ternary sequences are then summed into one larger ADD whose terminal nodes are normalized to ensure that the ADD represents a probability distribution. Thus, each inital ADD represents a probability array of size (n, m). The number of nodes of such an ADD is bounded from above by mn. After all the ADDs in the starting population are created, the variables $v_1 \cdots v_n$ are reordered by routines in the software package CUDD in view of minimizing the storage requirements of the entire set of ADDs. (The same variable ordering governs all the ADDs in the population.)

5.2.2 Crossover and mutation

One-point crossover between two ADDs A_1, A_2 proceeds as follows. Each truth assignment determines an binary number by associating truth with 1 and falsity with 0 (the ordering of variables determined by CUDD orders the digits of these numerals). One of these 2^n numbers is chosen uniformly randomly, and two distributions are created. In one distribution, the truth assignments below the chosen point are given probabilities according to A_1, the remaining truth assignments get probabilities according to A_2. The other distribution receives the complementary probabilities. In both cases, the resulting ADDs are renormalized to sum to unity. Mutation of a given ADD starts from another choice of a number between 1 and 2^n. With probability 0.5, the terminal nodes of all paths in the ADD that correspond to a truth assignment below the chosen number are multiplied by .9, and the remaining terminals are multiplied by 1.1. With probability 0.5, the terminal nodes of all paths in the ADD that correspond to a truth assignment below the chosen number are multiplied by 1.1, and the remaining terminals are multiplied by .9. The ADD is then renormalized.

5.2.3 Construction of successive generations

Let *Prob* represent the set of target estimates to be approximated, and let $Prob^*$ be the distribution represented by a given ADD A in our population. Define *dev* to be the absolute deviation between *Prob* and $Prob^*$ in the sense of (1)[11] We take the *fitness* of A to be $1/(dev + .01)$. Thus, greater match to *Prob* yields higher fitness. (Adding .01 prevents division by zero.) Between two generations, the probability of being selected for mating is proportional to fitness. For a population of (even) size N, $N/2$ pairs of ADDs are selected with replacement on this basis. With probability .95 the pair undergoes crossover. Whether crossed or not, the pair then undergoes mutation with probability .04. The two chromosomes then enter the next generation (again yielding a population of N). Better variable ordering is once again sought for the entire population.

[10] We did not achieve good results when ADDs were searched using simulated annealing. Similarly, searching through arrays with genetic algorithms performed poorly. We are unable to explain why certain combinations of data structures and search techniques work well whereas others do not.

[11] It can happen that there is a conditional event $(\varphi : \psi)$ among the target estimates for which the ADD's distribution $Prob^*$ is undefined (because $Prob^*(\psi) = 0$). In this case (which is extremely rare), we augment *dev* by a small value.

vars	30		40		50	
Inc	MAD	Min	MAD	Min	MAD	Min
0	.0037	56	.0029	106	.0033	127
.1	.0832	43	.0826	90	.0884	139
.2	.1687	26	.1699	46	.1711	75

Table 2: Mean absolute deviation (MAD) and time in minutes (Min) achieved using genetic algorithms and ADDs on simulated data generated from sparse distributions with 20,000 positive states. Times are relative to a Pentium III (533 MHz) processor running C code.

Across generations, the ADD with highest fitness (lowest *dev*) is retained, and its distribution is used to approximate *Prob*. We were prepared to discourage the growth of large ADDs by incorporating size into the measure of fitness. But we observed manageable growth of ADDs across generations, so *dev* alone was used to determine fitness.

5.3 Test of the new method

Let us use the abbreviation "GAADD" to denote the algorithm based on genetic algorithms and ADDs, and "SIMARR" to denote simulated annealing applied to arrays. We applied GAADD to the four experimental studies described in Sections 4.1 and 4.2 above. For each participant in the experiment on stock market forecasting we calculated the mean absolute deviation (MAD) between her 46 probability estimates and those offered by the best approximating ADD discovered for her. Across the 31 participants, the average MAD was .097, compared to .095 for SIMARR. Regarding the other experiments, the average MAD between a participant's estimates and those supplied by the best approximating ADD was was .085 for weather, .121 for Suns, and .094 for Mavericks. The MADs achieved with SIMARR were .085, .115, and .094, respectively. The quadratic scores for the coherent approximations delivered by GAADD were as low or lower than those for SIMARR.

The two methods thus constructed coherent approximations of similar quality. GAADD required more time than SIMARR, however. Whereas only a few minutes were required for SIMARR to complete its work on all the data in a given experiment, GAADD required hours.[12] A different picture emerges when GAADD is applied to the simulated data described in Section 4.3. The results are shown in Table 2, which can be interpreted analogously to Table 1 for SIMARR. GAADD here shows levels of performance that are slightly superior to those for SIMARR and in less time. The difference in the two comparisons between GAADD and SIMARR is likely connected to the format of the judgments in the two data sets. For the simulated data, we relied on disjunctive normal form, easily exploited by routines in the CUDD software package. Thus, computing the probabilities of formulas is much easier relative to ADDs than relative to probability arrays. In contrast, the judgments in the experimental data had a simpler structure. From these contrasts we surmise that in the present context simulated annealing is a faster search algorithm than genetic algorithms but that ADDs are more efficient than arrays for coding complex judgments.

[12]Moreover, GAADD was implemented in C whereas SIMARR relied on the JAVA virtual machine.

6 Discussion

We have described two methods for finding coherent approximations to incoherent probability estimates by human judges. The coherent approximations were shown to be objectively more accurate than the judges themselves. Furthermore, our algorithms provide a means to aggregate the opinions of distinct judges, and evidence was presented that enhanced accuracy is the result. The scalability of our methods was also demonstrated using synthetic data. We are currently testing scalability on real data by collecting probability estimates about stock and oil markets involving 30 variables. Algorithms are also being designed to exploit the factorization of a distribution that results from conditional independence among subsets of variables (Castillo et al., 1997). Such factorization might be built into the structure of a domain or else revealed by expert judgment.

Let us conclude by underlining the potential use of our technique for finding consensual agreement among experts. Each expert can offer estimates of events (or conditional events) that only partially overlap the events assessed by others. Provided that every pair of experts makes judgments over some shared variables, each expert will indirectly interact with every other (through the constraints imposed by the probability axioms on the coherence of the pooled estimates). The aggregation procedure can be iterated by allowing the experts to review and discuss the output of the algorithm, possibly modifying their original assessments at the end of each cycle. In this sense, our technique is an extension of traditional approaches to group consensus (see Adler & Ziglio, 1996). The algorithm can also be adapted to allow judges to attach varying levels of confidence to their estimates, and to allow higher authori-

ties to attach varying levels of confidence to judges (or even to specific judgments). It is worth noting that without any such modifications, our algorithm already respects pre-existing consensus in the following sense. Suppose that many judges assign nearly the same estimate to the chance of a specific event E. Then (under mild conditions) the algorithm will tend to change the estimate less than if the estimates for E were more varied.

Acknowledgements

The work of Osherson, Tsavachidis, and Vardi is partially supported by NSF grant IIS-9978135. We thank three anonymous reviewers for helpful comments on the first draft of this paper.

References

[1] M. Adler and E. Ziglio. *Gazing into the Oracle: The Delphi Method and its Application to Social Policy and Public Health.* Jessica Kingsley Publishers, Amsterdam, 1996.

[2] D. Aldous and U.V. Vazirani. "Go With the Winners" Algorithms. In *IEEE Symposium on Foundations of Computer Science*, pages 492–501, 1994.

[3] R. Bahar, E. Frohm, C. Gaona, G. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic decision diagrams and their applications. *Journal of Formal Methods in Systems Design*, 10(2/3):171–206, 1997.

[4] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, c-35(8), 1986.

[5] E. Castillo, J. Gutiérrez, and A. Hadi. *Expert systems and probabilistic network models.* Springer, New York, 1997.

[6] V. Chvátal. *Linear Programming.* W. H. Freeman, San Francisco CA, 1983.

[7] M. J. Druzdzel and L. C. van der Gaag. Elicitation of probabilities for belief networks: combining qualitative and quantitative information. In *Proc. 11th Conference on Uncertainty in Artificial Intelligence*, pages 141–148, Los Altos, CA, 1995. Morgan Kaufmann Publishers.

[8] R. Fagin, J. Halpern, and N. Megiddo. A Logic for Reasoning about Probabilities. *Information and Computation*, 87:78 – 128, 1990.

[9] G. Georgakopoulos, D. Kavvadias, and C. Papadimitriou. Probabilistic satisfiability. *Journal of Complexity*, 4:1–11, 1988.

[10] M. Mitchell. *An introduction to genetic algorithms.* MIT Press, Cambridge MA, 1996.

[11] D. Osherson. Probability judgment. In E.E. Smith and D. Osherson, editors, *Invitation to Cognitive Science: Thinking (2nd Edition).* M.I.T. Press, Cambridge MA, 1995.

[12] A. Tversky and D. Kahneman. Extensional versus intuitive reasoning: The conjunction fallacy in probability judgment. *Psychological Review*, 90:293–315, 1983.

[13] L. van der Gaag, S. Renooij, C. Witteman, B. Aleman, and B. Taal. How to elicit many probabilities. In K. B. Laskey and H. Prade, editors, *Proc. 15th Conference on Uncertainty in Artificial Intelligence*, pages 647–654, Los Altos, CA, 1999. Morgan Kaufmann Publishers.

[14] P. van Laarhoven. *Theoretical and computational aspects of simulated annealing.* Center for Mathematics and Computer Science, Amsterdam, 1988.

[15] R. Viale and D. Osherson. The Diversity Principle and the Little Scientist Hypothesis. *Foundations of Science*, 5:239–253, 2000.

[16] D. von Winterfeldt and W. Edwards. *Decision analysis and behavioral research.* Cambridge University Press, New York NY, 1986.

[17] J. F. Yates. *Judgment and Decision Making.* Prentice Hall, Englewood Cliffs NJ, 1990.

Evolving Ontologies in Distributed and Dynamic Settings

Helena Sofia Pinto
Grupo de Inteligência Artificial
Departamento de Eng. Informática
Instituto Superior Técnico
Av. Rovisco Pais, 1049-001 Lisboa, Portugal

João P. Martins
Grupo de Inteligência Artificial
Departamento de Eng. Informática
Instituto Superior Técnico
Av. Rovisco Pais, 1049-001 Lisboa, Portugal

Abstract

The Semantic Web provides an enormous source of knowledge. Its development is both distributed and highly dynamic. New technology is needed to cope with such "lively" environment. This report from the field describes an experience in ontology development that is both distributed and dynamic: an ontology reuse experience with an evolving ontology, making it necessary to apply and reapply an ontology integration process. We believe that this experience can give us invaluable insight as to the issues that have to be dealt with in distributed and dynamic processes. We believe that the technology associated to the Semantic Web will have to deal with the issues reported in this article. Moreover, since ontologies are an essential component of the Semantic Web it is important to understand the new ways in which they will be built.

1 INTRODUCTION

Today the Semantic Web provides an enormous source of knowledge. The development of this source is both (1) distributed, anyone can contribute with more knowledge, and (2) highly dynamic, several contributors may be adding knowledge at the same time. New technology is needed to cope with such "lively" environment.

So far, ontology development experiences were performed either in a distributed, but controlled way or in a centralized way. In this paper we report an experience in ontology development that followed a distributed, dynamic process: an ontology reuse experience involving an evolving ontology, making it necessary to apply and reapply an ontology integration process (Pinto, Gómez-Pérez & Martins 1999). We believe that this experience can give us invaluable insight as to the issues that have to be dealt with in distributed and dynamic processes. We need to understand these kind of processes so that the new technology to be developed is able to deal with such problems. We believe that the technology associated to the Semantic Web will have to deal with the issues reported in this article. Moreover, since ontologies are an essential component of the Semantic Web (Fensel & Musen 2001) it is important to understand the new ways in which they will be built.

In this article we begin by describing the main features of ontology development. Then, we describe the process of reusing a "subontology" about Units of Measure, which was a part of a more general upper ontology. The upper ontology was being built at the same time that its "subontology" was being reused. Due to the reuse process that was being applied to the "subontology", the upper ontology also evolved.

2 BUILDING ONTOLOGIES

Ontology building is a process composed of several activities, Figure 1. Some are performed at particular stages: specification, conceptualization, formalization, implementation and maintenance. Others take place along the entire life cycle: knowledge acquisition, documentation and evaluation.

It is more or less consensual in the Ontological Engineering field that the development of an ontology follows an evolving prototyping life cycle,[1] rather than a waterfall (Royce 1970) or an iterative one (Basili & Turner 1995), Figure 2. In an evolving prototyping life cycle, one can go back from any stage to any previous stage of the development process. As long as the ontol-

[1] Also called evolutionary.

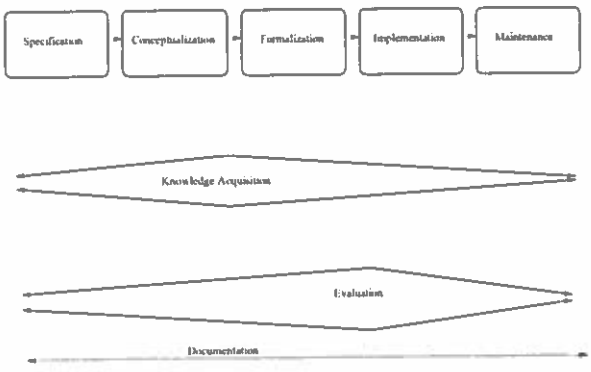


Figure 1: Activities in the ontology development life cycle

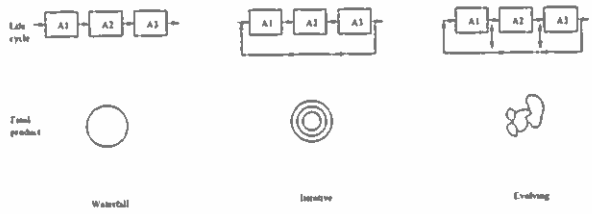


Figure 2: Comparison of life cycles

ogy being built does not satisfy evaluation criteria and does not meet all requirements found during specification, the prototype is improved. One should note that evaluation should be performed along the whole life cycle, although some ontology building methodologies have a specific evaluation stage after implementation. Usually, one does not start over a new prototype in each "iteration", it only improves the existing one.[2] Any part of the ontology that was identified as lacking quality or not meeting the desired requirements is improved.

Ontologies can be build from scratch or by reusing existing ontologies. Regarding ontology reuse, there are two different processes (Pinto et al. 1999): integration/composition and merge/fusion. Integration/composition is the process of building an ontology in one subject reusing one or more ontologies in different subjects (the subjects of the different ontologies may be related) (Pinto et al. 1999). In an integration process, source ontologies are aggregated, combined, assembled together, to form the resulting ontology, possibly after reused ontologies have suffered some changes, such as, extension, specialization or adaptation. Source ontologies are part of the resulting ontology (the resulting ontology may be composed of several modules, "subontologies"). In an integration process one can identify in the resulting ontology regions that were taken from the integrated ontologies. Knowledge in those regions was left more or less unchanged, Figure 3. It should be noted that the integration process is included in the overall process of ontology building.

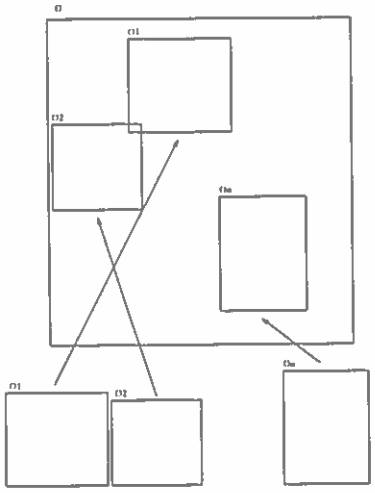


Figure 3: Integration/composition

The merge/fusion process is the other ontology reuse process. Merge is the process of building an ontology in one subject reusing two or more different ontologies on that subject (Pinto et al. 1999). In a merge process source ontologies are unified into a single one, so it usually is difficult to identify regions in the resulting ontology that were taken from the merged ontologies and that were left more or less unchanged. Knowledge from the source ontologies is homogenized and altered through the influence of one source ontology on another (in spite of the fact that the source ontologies do influence the knowledge represented in the resulting ontology). In some cases, the knowledge from one particular source ontology is scattered and mingled with the knowledge that comes from the other sources, Figure 4. It should be stressed that in a merge process source ontologies are truly different ontologies and not simple revisions, improvements or variations of the same ontology.

So far, the experiences in ontology development were performed either in a distributed, but controlled way (for instance, (KA)^2 (Benjamins & Fensel 1998)) or in a centralized way (for instance, Chemicals (Fernández, Gómez-Pérez, Sierra & Sierra 1999)). By distributed but controlled way we mean that:

[2] Unless the existing prototype was found completely inadequate or the cost of modifying it outweighs the cost of building a new one.

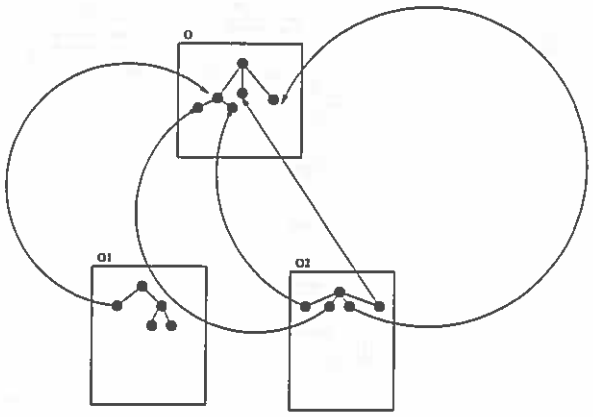


Figure 4: Merge/fusion

- there is a group of people in charge of each part of the ontology that decides upon the knowledge that should be represented in that part,
- there is a group controlling the overall effort that manages and coordinates the work performed by each group.

Moreover, usually the overall group involved in the effort is or can be known. The only exception, as far as we know, reported in the literature is the Reference ontology (Arpirez-Vega, Gomez-Perez, Lozano-Tello & Pinto 2000), an ontology about ontologies. Its development took a bipartite form. The development of the conceptual structure of the ontology was centralized (the developers of the Reference ontology decided which features should be used to characterize ontologies). The effort made to collect information about ontology instances was distributed among ontology developers but controlled since a rigid framework was provided and had to be followed (a questionnaire that guides which information should be provided for each instance).

In the Semantic Web the introduction of new information follows an even more distributed and uncontrolled way:

- anyone can contribute with more knowledge,
- several contributors may be adding knowledge at the same time,
- there is no control over what information is added, and the quality and reliability of that information.

We need to understand these kind of processes so that the new technology to be developed copes with such kind of environments.

3 AN EVOLVING ONTOLOGY: THE UNITS OF MEASURE "subontology" OF SUMO

In the beginning of 2001 an initial version of the SUMO ontology (Niles & Pease 2001) was sent to the all members of the IEEE P1600.1 Standard Upper Ontology working group for comments and revisions. Shortly after that we began an integration process reusing part of this ontology.

A part of the SUMO ontology is about Units of Measure. The initial version of this part of the ontology was built using the Standard Units ontology (Gruber & Olsen 1994) on the Ontolingua Server[3] (Farquhar, Fikes & Rice 1996).

In our integration experience, we needed an ontology about this domain, so we decided to reuse this part of SUMO. The motivations for this integration effort were not only revise and improve the proposed SUMO ontology, which is being built, but also perform another integration experience to test the methodology (Pinto & Martins 2001a, Pinto 2000, Pinto 1999) that we developed to help and guide the integration process (Pinto & Martins 2001a, Pinto 2000, Pinto et al. 1999).

An important activity in an integration process is the analysis of candidate ontologies from both a technical and an user point of view: ontologies should be technically evaluated by domain experts from a domain point of view and user assessed by ontologists from a knowledge representation point of view using specialized sets of criteria oriented to integration. This analysis assures that the ontologies that are integrated have enough technical quality and are useful to build the resulting ontology. Since this analysis identifies the weaknesses and strong points of candidate ontologies (most of the times, the ontology to be reused is not a perfect candidate, therefore we must identify what must be changed) and since a subsequent activity in the integration process is the application of integration operations, that transform source ontologies into "perfect"[4] candidates, the integration process entails the improvement of the technical quality of source ontologies.[5]

[3] http://WWW-KSL-SVC.stanford.edu:5915

[4] For that particular reuse process.

[5] It also improves their usefulness for the particular integration process that is being applied.

3.1 THE REVISION/INTEGRATION PROCESSES

We wanted to follow a principled approach in this revision in order to have some guarantees that the changes to be proposed would effectively improve the technical quality of the ontology and would contribute to increase the confidence of future SUMO users. There are several criteria to analyze ontologies: design criteria (Gruber 1995), general evaluation and assessment criteria (Gómez-Pérez, Juristo & Pazos 1995, Gómez-Pérez 1999), general evaluation criteria for general reuse (Gómez-Pérez & Rojas-Amaya 1999) and specific integration-oriented evaluation and assessment criteria (Pinto & Martins 2000, Pinto 1999), that, in general, subsume the previous ones. Since the last set of criteria is specific for integration, which will be one of the main processes involved in reusing the SUMO ontology, and since we were performing an integration process we chose it. We also believe that by following these criteria while building the ontology, improves its reusability and usability.

In our methodology we recommend that integration should begin as early as possible, preferably during conceptualization, so that the overall effort of building the ontology is lessened. At this early stage one uses knowledge level (Newell 1982) representations of ontologies, conceptual models. Moreover, since domain experts usually are not familiar with knowledge representation languages they are not able to understand the implemented version of an ontology (Fernández et al. 1999). Since they have to evaluate the ontologies, they should be given their knowledge level representations and not their implemented versions. If the knowledge level representation of an ontology is not available we must perform a reengineering process (Gómez-Pérez & Rojas-Amaya 1999) to get a conceptual model of the ontology from its implementation. Therefore, by avoiding the reengineering process we can lessen the overall integration effort. The SUMO ontology is only available at the implementation level. In the particular integration process that we applied to the Units of Measure "subontology" of SUMO, the domain experts were also the ontologists. Since, in this case the part of the ontology being reused is not very large (not many concepts), is hierarchically organized as a tree (not a tangled hierarchy) and this tree is not very deep, the ontologists could easily abstract the conceptual model of this part of the ontology directly from its implementation. Therefore, this particular integration process was performed at the implementation level.

The activities in an integration process are represented in Figure 5 (Pinto & Martins 2001a, Pinto 2000). Recall that integration is included in the overall ontology building process.

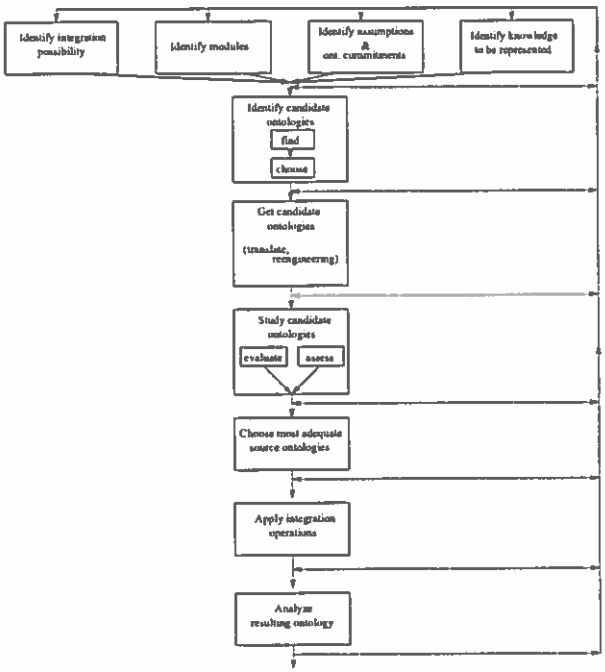


Figure 5: The integration process

The revision/integration process followed the following *steps*:

1. An initial iteration of the revision/integration process was performed to the Units of Measure "subontology" of the SUMO ontology. The result was a first new version for this part of SUMO, Figure 6.

2. Once the first iteration of the revision/integration of the Units of Measure "subontology" of SUMO was performed we sent it to the SUMO developers. Their reaction was extremely positive due to the increase in technical quality that had been achieved. Then, the first new version of the Units of Measure "subontology" was incorporated in the SUMO ontology by its developers, Figure 6. However, in the meantime the whole SUMO ontology, in particular the Units of Measure "subontology", had also evolved. This was due to the fact that knowledge from other sources had been merged into this part of SUMO, namely knowledge from the Quantities ontology developed by ITBM-CNR[6]. Therefore, after the incorporation

[6] `http://saussure.irmkant.rm.cnr.it/onto`

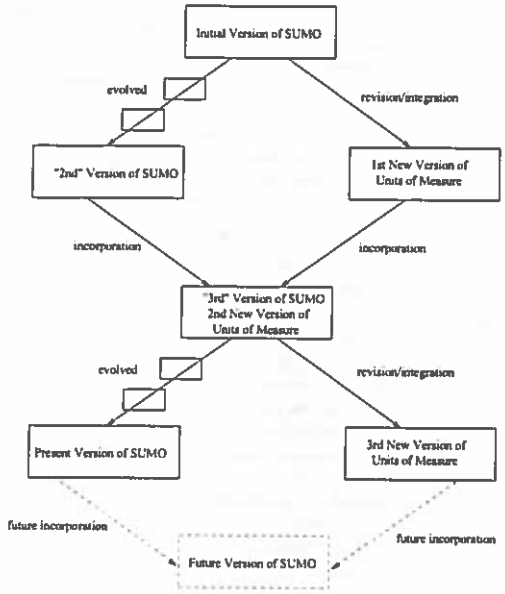


Figure 6: Evolution of the Units of Measure "subontology"

of the first new version of the "subontology" in the main version of the SUMO ontology, a second new version for this "subontology" was created. It combined the changes introduced by the first revision/integration process and the changes introduced in the evolving SUMO ontology.

3. Since the source ontology of our integration process had in the meanwhile evolved, we had to reapply the revision/integration process to the second new version of the Units of Measure "subontology" of SUMO (version 1.15, June 2001). As a result of this process a third new version for the Units of Measure "subontology" is being built, Figure 6.

In the future, more iterations of the revision/integration process may have to be performed until a more or less stable version for the Units of Measure "subontology" is found.

In the next sections we describe the most important aspects of each one of the steps followed.

3.1.1 First Iteration of Revision/Integration

The *steps* followed in the first iteration of the revision/integration process were:

1. We began the analysis of the Units of Measure "subontology" using the criteria proposed in (Pinto & Martins 2000, Pinto 1999). We had already some knowledge about the domain[7] so we could begin the analysis before acquiring some knowledge about the domain.

2. Since we detected some problems we did some knowledge acquisition. We mainly looked into on-line sources of knowledge to speed up the knowledge acquisition process. The best source of knowledge found was the NIST Reference[8] on Constants, Units, and Uncertainty. We also used the on-line version of the Encyclopaedia Britannica. While choosing on-line sources special care was taken to choose the most reputable ones. One of the problems of using on-line sources of knowledge is the quality of those sources. If one wants to build an ontology, knowledge should be acquired from the most reputable sources as possible, such as (*Le Systéme International d'Unités (SI), The International System of Units (SI)* 1991) or (Taylor 1995) for the Standard Units domain. Only if this policy is followed does the ontology provide quality guarantees to its reusers/users.

3. Then, we completed the analysis of the ontology following the aforementioned criteria, namely we verified every conversion function.[9]

4. Then, we implemented all proposed changes in the ontology.[10] At the same time that we changed the ontology to eliminate identified problems, minor mistakes were found and corrected.

5. After implementing all changes, we analyzed the quality of the resulting ontology using the criteria proposed in (Pinto & Martins 2001a, Pinto 2000).

6. Since some minor mistakes were detected, they were corrected. The desired Units of Measure "subontology" had been built.

The *results* of the initial *technical evaluation and user assessment* of the Units of Measure "subontology" are described in (Pinto & Martins 2001b). As a result of this analysis several changes were applied to the

[7] We have Mechanical Engineering backgrounds and some knowledge in the Chemistry, Thermodynamics and Electrical domains, so we are familiar with most units appearing in the SI system of units and have extensively manipulated SI units.

[8] `http://physlab.nist.gov/cuu/Units/index.html`

[9] A function that allows us to convert values expressed in a given unit of measure into another unit for the same physical quantity.

[10] This corresponds to the application of integration operations.

ontology. These changes correspond to the application of *integration operations* (Pinto 2000, Pinto & Martins 2001a) that transform the reused version of the ontology into the appropriate ontology to the integration process at hand. The main integration operations that were applied were:

correction of definitions The definitions of a few knowledge pieces represented in the ontology had to be corrected.

change of knowledge sources This operation (Pinto 2000) entailed a vast revision of the ontology, namely:

- Extensive improvement and introduction of the documentation in the ontology, since one of its goals is teaching the domain to non-experts.
- Correction of the definitions of the knowledge pieces represented in the ontology. All conversion functions were made to convert into the SI base unit or the SI derived unit with special name and symbol for that physical quantity and not to its multiples or submultiples. All conversion factors were verified and changed, if necessary, into more accurate ones.
- Introduction of SI standard terminology. Not only this new terminology is the standard, but also it follows the accepted rules and style conventions proposed by the standard.
- Introduction of relevant knowledge pieces that were missing. For instance, the classification of the physical quantity magnetic flux measure, the SI unit to measure this physical quantity (weber), the assertions about which units are SI units, the missing conversion functions, etc. In fact, the changes to be performed were so extensive that is also entailed the application of another operation: **extend** (Pinto 2000).
- Removal of all non-relevant knowledge pieces. In fact, the changes to be performed were so extensive that is also entailed the application of another operation: **cut** (Pinto 2000).

The *design criteria* that were used to guide the application of integration operations were (Arpirez-Vega et al. 2000): modularize, specialize, diversify each hierarchy, minimize the semantic distance between sibling concepts, maximize relationships between taxonomies and standardize names of relations. In this particular case an additional criterion was also used for this step

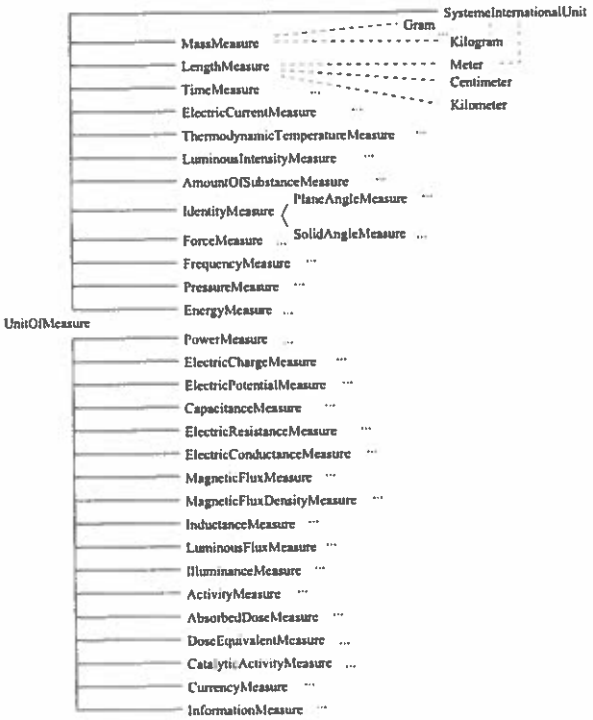


Figure 7: Units of Measure "subontology" after first revision/integration iteration

of the process, so that the new version of the Units of Measure "subontology" of SUMO, or at least some parts of it, could more easily be latter introduced back in the ontology:

- Modifications in its structure should be kept to a *minimum*. To comply to this criterion it was decided that at this first stage we should not introduce the distinction between base units, derived units and derived units with special names and symbols.

In Figure 7 we represent part of the structure of the Units of Measure "subontology" of SUMO after the first revision/integration iteration. Solid lines represent class-subclass relationships and dashed lines represent class-instance relationships.

3.1.2 Incorporation Step

The incorporation step was performed by SUMO developers. This process should not be confused with a merge/fusion process since the ontologies involved are not not truly different ontologies. They were improvements or variations of the same ontology.

The important point is the fact that we were not in-

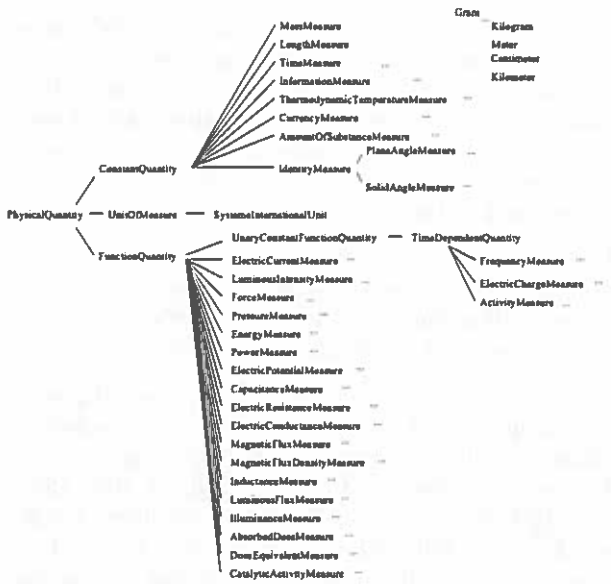


Figure 8: Units of Measure "subontology" after incorporation

volved either in this part of the process or in the evolution that the SUMO ontology had suffered while we were performing the first iteration of the revision/integration process. The ontology changed dynamically and in a completely distributed way.

In Figure 8 we represent part of the structure of the Units of Measure "subontology" of SUMO after the incorporation step.

3.1.3 Second Iteration of Revision/Integration

Once we received the version of the SUMO ontology incorporating the proposed changes we performed another revision/integration iteration.

It is important to refer that some of the changes proposed in the first revision/integration iteration (Pinto & Martins 2001b) had not been performed. Some of them involved changes in the basic distinctions represented in the ontology, that usually introduce changes in the structure of the ontology. The changes that were not introduced are described in the presentation of (Pinto & Martins 2001b) kept in `http://reliant.teknowledge.com/IJCAI01/`. In this iteration we wanted to introduce them.

The *steps* followed, so far, in the second iteration of this revision/integration were:

1. Once again, we analyzed the Units of Measure "subontology" using the criteria proposed in (Pinto & Martins 2000, Pinto 1999). Two major concerns in this analysis were: (1) whether all the changes that had been sent to SUMO developers had been integrated and (2) which new changes had in the meanwhile been introduced. We needed to know how different the new version of the "subontology" was from the version that had been sent to SUMO developers. We found that some of the changes proposed by us had not been introduced. These cases are described in the presentation of (Pinto & Martins 2001b). However, in the meanwhile, it has been agreed between the two parties that they should be introduced. Some of the changes introduced by SUMO developers that were not found important by us were also discussed between the two parties. Communication between the parties involved in the development of an evolving ontology is essential if a common shared ontology that can be agreed by the largest number of entities is sought.

2. Since the new version had introduced some changes in the structure of the ontology and since we already wanted to introduce changes at that level, a new structure was sought for the upper-levels of the Units of Measure "subontology" that could reach the agreement of all parties involved in its development.

3. At this moment, the new implementation including all changes is under construction.

The main *integration operations* that are being applied at this moment are:

introduction of knowledge pieces Introduction of relevant knowledge pieces of the domain that are missing.[11]

restructuring Some of the distinctions that are important to accurately describe the domain are being introduced, namely the distinctions of base units, derived units, derived units with special names and symbols.

Regarding the integration process that is being applied to reuse this "subontology" we found that: (1) some of the operations no longer had to be applied because the changes we had proposed were introduced into SUMO, (2) others still have to be applied because the corresponding changes have not been introduced, and (3) new operations have to be applied since unexpected,

[11] Classes, instances, relations, functions, etc.

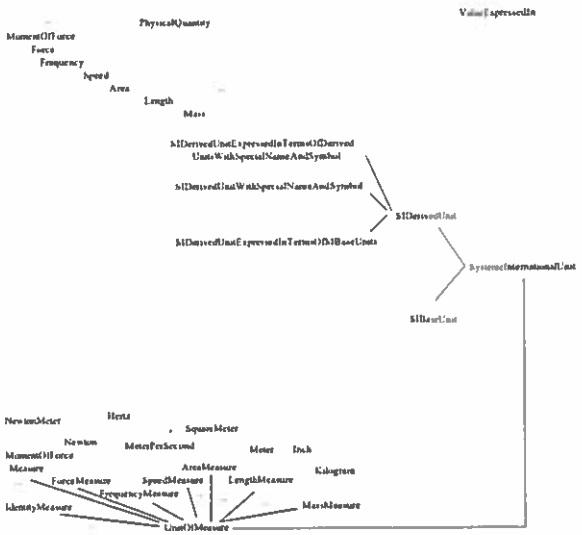


Figure 9: Units of Measure "subontology" after second revision/integration iteration

and unneeded changes (for the particular reuse process that is being applied to this "subontology") were introduced in the ontology. For this reason, the log of integration operations that is kept (Pinto 2000) proved essential to ease the reapplication of the integration process.[12]

In Figure 9 we represent part of the structure of the Units of Measure "subontology" of SUMO after the second revision/integration iteration. The dotted line represents the binary relation "ValueExpressedIn" that relates physical quantities with the corresponding units of measure. This process is still being performed.

4 DISCUSSION

Ontologies usually are either built in a centralized or in a distributed, but controlled way. The development of the Semantic Web, for which ontologies are an essential component, is both distributed and highly dynamic. We have been involved in the development of an ontology that followed an intermediate kind of process: distributed and dynamic. The ontology is being simultaneously built by several parties that collaborate in the task.

One of the interesting features of the process reported in this article is the fact that it was a true evolving prototyping process. In fact there were several independent parties changing the ontology at the same time. Each party had no control over the changes that were being performed by the other parties. Therefore, at some point the need arose to bring together the different versions of the "subontology".

In some cases the different versions of the ontologies may grow as independent variants. However, the effort to bring them together should be sought if it entails the improvement of its technical quality.

There is an urgent need for ontology building tools that keep track of the several versions of an ontology. Existing environments such as the Ontolingua Server, Ontosaurus[13] (Swartout, Patil, Knight & Russ 1997) or PROTÉGÉ (Noy, Sintek, Decker, Crubézy, Fergerson & Musen 2001) do not deal with this issue. Moreover, it would be useful if these tools had mechanisms that direct users to the points where the different versions of the ontologies differ.

The ontology revision/integration processes described in this article were all manually performed. Ontology building tools that help perform reuse processes are needed, namely integration and merge. It is important that these tools help the process of reapplying reuse processes. This need arises from the fact that source ontologies change, evolve, making it necessary to reapply the process. Since reuse processes are rather complex it is desirable that, whenever they are performed, some form is found of simplifying their reapplications.

For the effective and useful use of the Semantic Web ontologies are needed. For instance, ontologies will allow the development of more intelligent and precise information retrieval services. However, it is not foreseeable that a single ontology can be built for such an evolving environment. Therefore, we must understand the processes and problems behind highly dynamic and distributed environments. Although our experience is distributed and dynamic it was more controlled than in case of the Semantic Web: all parties involved in the experience were focused on the same task, had the same purpose[14] and were contributing in a positive way for the same effort. During the whole revision/integration process special care was taken with the quality of the knowledge sources that were chosen to be used to correct the ontology. In general, in the Semantic Web this cannot be assured because there is a lot of noise: (1) positive and negative contributions that lead to contradictions that need to be solved and

[12] This log should only be used as a guideline. When reapplying integration, the transformations that were performed to the source ontology may have eliminated the need for the application of some integration operations or may have created the need for more integration operations to be performed.

[13] http://www.isi.edu/isd/ontosaurus.html

[14] Build an upper ontology. See http://suo.ieee.org/

(2) lack of information about the reliability and quality of the information being provided by the several parties.

5 CONCLUSION AND FUTURE WORK

The experience reported in this article is particularly interesting because the development of the ontology followed a true evolving prototyping process. It took place in a dynamic and distributed way involving different groups working at the same time in the same product. Special care was given to the quality of the product being built in every stage of the process: special care choosing the sources of knowledge from which to acquire knowledge, several technical evaluations and user assessments of the ontologies to be reused and of the final product.

The SUMO ontology is still evolving. It will probably take a few more iterations before a stable version is found. For instance, it should be extended by the inclusion of the concept of precision. When we measure some physical quantity there is always an error associated to the measurement act. In the future it will be probably divided into several subontologies one for each system of units (there are other systems of units besides the SI system, for instance CGS).

Ontology building methodologies for the development of ontologies in distributed and dynamic settings are needed. Further experiences must be conducted before they can be proposed.

Acknowledgements

We would like to thank the anonymous reviewers for their comments.

References

Arpirez-Vega, J., Gomez-Perez, A., Lozano-Tello, A. & Pinto, H. S. (2000). Reference Ontology and (ONTO)^{2}Agent: the Ontology Yellow Pages, *Knowledge and Information Systems* **2**(4): 387–412.

Basili, V. & Turner, A. (1995). Iterative Enhancement, a practical technique for software development, *IEEE Transactions on Software Engineering* **1**(4).

Benjamins, R. & Fensel, D. (1998). The Ontological Engineering Initiative (KA)^2, *in* N. Guarino (ed.), *Formal Ontology in Information Systems*, IOS Press, pp. 287–301.

Farquhar, A., Fikes, R. & Rice, J. (1996). The Ontolingua Server: A Tool for Collaborative Ontology Construction, *Proceedings of the Knowledge Acquisition Workshop, KAW96.*

Fensel, D. & Musen, M. A. (2001). The Semantic Web, *IEEE Intelligent Systems* **16**(2): 24–25.

Fernández, M., Gómez-Pérez, A., Sierra, A. P. & Sierra, J. P. (1999). Building a Chemical Ontology Using METHONTOLOGY and the Ontology Design Environment, *IEEE Expert (Intelligent Systems and Their Applications)* **14**(1): 37–46.

Gómez-Pérez, A. (1999). Evaluation of Taxonomic Knowledge in Ontologies and Knowledge Bases, *Proceedings of the Knowledge Acquisition Workshop, KAW99.*

Gómez-Pérez, A., Juristo, N. & Pazos, J. (1995). Evaluation and Assessment of the Knowledge Sharing Technology, *in* N. Mars (ed.), *Towards Very Large Knowledge Bases*, IOS Press, pp. 289–296.

Gómez-Pérez, A. & Rojas-Amaya, D. (1999). Ontological Reengineering for Reuse, *in* D. Fensel & R. Studer (eds), *Proceedings of the European Knowledge Acquisition Workshop, EKAW99*, Springer Verlag.

Gruber, T. (1995). Towards Principles for the Design of Ontologies for Knowledge Sharing, *International Journal of Human Computer Studies* **43**(5/6): 907–928.

Gruber, T. & Olsen, G. R. (1994). An Ontology for Engineering Mathematics, *in* J. Doyle, E. Sandewall & P. Torasso (eds), *KR94 Proceedings*, Morgan Kaufmann, pp. 258–269.

Le Systéme International d'Unités (SI), The International System of Units (SI) (1991). Bur. Intl Poids et Mesures, Sévres, France. 6th Edition.

Newell, A. (1982). The Knowledge Level, *Artificial Intelligence* **18**(1): 87–127.

Niles, I. & Pease, A. (2001). Origins of the Standard Upper Merged Ontology: A Proposal for the IEEE Standard Upper Ontology, *Proceedings of IJCAI2001's Workshop on IEEE Standard Upper Ontology.*

Noy, N. F., Sintek, M., Decker, S., Crubézy, M., Fergerson, R. W. & Musen, M. A. (2001). Creating Semantic Web Contents with PROTÉGÉ-2000, *IEEE Intelligent Systems* **16**(2): 60–71.

Pinto, H. S. (1999). Towards Ontology Reuse, *Proceedings of AAAI99's Workshop on Ontology Management, WS-99-13*, AAAI Press, pp. 67–73.

Pinto, H. S. (2000). *Ontology integration: Characterization of the Process and a Methodology to Perform it*, PhD thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa. Discussed in 2001.

Pinto, H. S., Gómez-Pérez, A. & Martins, J. P. (1999). Some Issues on Ontology Integration, *Proceedings of IJCAI99's Workshop on Ontologies and Problem Solving Methods: Lessons Learned and Future Trends*, pp. 7.1–7.12.

Pinto, H. S. & Martins, J. (2000). Reusing Ontologies, *Proceedings of AAAI 2000 Spring Symposium Series, Workshop on Bringing Knowledge to Business Processes, SS-00-03*, AAAI Press, pp. 77–84.

Pinto, H. S. & Martins, J. (2001a). A Methodology for Ontology Integration, *Proceedings of the First Internation Conference on Knowledge Capture (K-CAP2001)*.

Pinto, H. S. & Martins, J. (2001b). Revising and extending the Units of Measure "subontology", *Proceedings of IJCAI2001's Workshop on IEEE Standard Upper Ontology*.

Royce, W. (1970). Managing the development of large software systems: concepts and techniques, *Proceedings of IEEE WESTCON*, pp. 1–9.

Swartout, B., Patil, R., Knight, K. & Russ, T. (1997). Toward Distributed Use of Large-Scale Ontologies, *Proceedings of AAAI97 Spring Symposium Series, Workshop on Ontological Engineering*, pp. 138–148.

Taylor, B. N. (1995). Guide for the Use of the International System of Units (SI), United States Department of Commerce, National Institute of Standards and Technology. NIST Special Publication 811.

Social contraction and belief negotiation

Richard Booth
University of Leipzig
Department of Computer Science
Augustusplatz 10/11
04109 Leipzig, Germany
booth@informatik.uni-leipzig.de

Abstract

An intelligent agent may receive information about its environment from several different sources. How should the agent merge these items of information into a single, consistent piece? Taking our lead from the contraction + expansion approach to belief revision, we envisage a two-stage approach to this problem. The first stage consists of weakening the individual pieces of information into a form in which they can be consistently added together. The second, trivial, stage then consists of simply adding together the information thus obtained. This paper is devoted mainly to the first stage of this process, which we call *social contraction*. We consider both a postulational and a procedural approach to social contraction. The latter builds on the framework of *belief negotiation models* (Booth 2001). With the help of Spohn-type rankings we provide two possible instantiations of this extended framework. This leads to two interesting concrete families of social contraction functions.

1 INTRODUCTION AND PRELIMINARIES

An intelligent agent may receive information about its environment from several different sources. How should the agent *merge* these pieces of information into a single, consistent piece? This question has recently received various treatments (see, for e.g., (Booth 2001, Cantwell 1998, Konieczny & Pino-Pérez 1998, Konieczny & Pino-Pérez 1999, Liberatore & Schaerf 1998, Maynard-Reid & Shoham 2001, Meyer 2001, Revesz 1993)). The simplest thing to do would be to just take the given pieces of information and *conjoin* them. While this strategy would be fine if the pieces of information are jointly consistent, it could well be that some of the pieces stand in contradiction, in which case the strategy breaks down. In this paper we envisage a two-stage approach to the problem: (i) the individual, raw pieces of information are manipulated (more precisely, *weakened*) into a form in which they *become* jointly consistent, and *then* (ii) the pieces thus obtained are conjoined. Stage (ii) is trivial. Stage (i) is not, and so forms the main topic of this paper.

A precedent for this two-stage approach can be found in the literature on the closely-related area of *belief revision* (Alchourrón et al. 1985, Gärdenfors 1988, Hansson 1999). Belief revision may essentially be thought of as "binary merging". It addresses the problem of how to merge one item of information, usually taken to represent the current beliefs of some agent, with another item, representing some new piece of information which the agent acquires. The idea, which dates back to (Levi 1977) and is given succinct expression by the *Levi Identity* (Gärdenfors 1988), is that this operation of revision is decomposed into two sub-operations: (i) *contraction*: the current information is weakened so that it becomes consistent with the new information, then (ii) *expansion*: the new information is simply added to the result. Note that, in (i), only the current information is weakened, not the new. This reflects the traditional assumption that the new information is always *completely reliable*. What we seek in this paper is a generalised version of the contraction operation, in which several items of information may all be weakened simultaneously so that they become consistent with one another. For this reason we call the operations we are interested in *social contraction* functions (SC functions for short).

We shall examine social contraction from two viewpoints: a *postulational* one and a more *procedural* one. For the latter we build on the framework of *belief nego-*

tiation models, which was introduced in (Booth 2001) as a framework for binary merging in which the merging is achieved via a negotiation-like process. We extend this framework so that it can handle information coming from n sources for $n \in \mathbb{N}$, and show how a given belief negotiation model yields an SC function.

The plan of the paper is as follows. We begin in Section 2 by formally defining SC functions via a small list of basic properties we expect such an operation to satisfy. We show how one of these basic properties allows us to derive, from a given *social* contraction function, a list of *individual* contraction functions (in the traditional belief revision sense as described above) — one for each information source. We also describe how a given SC function yields a merging operator via a kind of "generalised" Levi Identity before ending the section with a look at a few possible additional postulates for social contraction, relating to the idea — familiar from belief revision — of *minimal change*. The rest of the paper is devoted to belief negotiation. The extended framework is set down in Section 3, where it is shown how each (extended) belief negotiation model yields an SC function and, conversely, how every SC function can be said to arise in this way. As we will see, the framework is set at a very abstract level. Section 4 is all about putting a little more flesh on the bones. Making heavy use of Spohn-type rankings we provide two, intuitively plausible, instantiations of the parameters of a belief negotiation model, giving in the process two concrete families of SC functions. We characterise the behaviour of the individual contraction functions as well as the merging operators which are derivable from these particular families. It turns out that they are all familiar from the literature. We thus give a new angle on these operators by providing new "negotiation-style" characterisations for them. We also test the SC functions from each of these two families against the extra minimal change postulates from Section 2. We will see that the SC functions from the second family fare better than those from the first in this regard. We conclude in Section 5.

Preliminaries: We let $\mathcal{W}$ be the (finite) set of worlds, i.e., truth-assignments, associated with some fixed background propositional language generated from finitely many propositional variables. The set of all non-empty subsets of $\mathcal{W}$ we denote by $\mathcal{B}$. Given $S \subseteq \mathcal{W}$, we use $\overline{S}$ to denote $\mathcal{W} - S$. We assume throughout that we have a fixed finite set $Sources = \{1, \ldots, n\}$ of information sources ($n \geq 2$). We work semantically throughout, so each item of information provided by a source i will take the form of a set $S_i \in \mathcal{B}$ (so no source ever provides the "inconsistent" information $\emptyset$). An *information profile* (relative to $Sources$) is an element of $\mathcal{B}^n$. We use $\vec{S}, \vec{S}^1$, etc. to denote information profiles, with S_i, S_i^1, etc. denoting the i^{th} element of $\vec{S}, \vec{S}^1$, etc. The idea is that S_i is the information in $\vec{S}$ belonging to source i. We will say an information profile $\vec{S}$ is *consistent* when $\bigcap_i S_i \neq \emptyset$, otherwise it is *inconsistent*. Given two information profiles $\vec{S}^1$ and $\vec{S}^2$, we will write $\vec{S}^1 \subseteq \vec{S}^2$ to mean $S_i^1 \subseteq S_i^2$ for all $i \in Sources$. Finally if $\mathbf{f}$ is a function with range $\mathcal{B}^n$, we will use $\mathbf{f}_i(\vec{S})$ to denote the i^{th} element of $\mathbf{f}(\vec{S})$.

2 SOCIAL CONTRACTION FUNCTIONS

Our first aim is to get a formal definition of SC functions up and running. Intuitively we want an SC function to be a function $\mathbf{f} : \mathcal{B}^n \to \mathcal{B}^n$ which, given an information profile $\vec{S}$ provided by $Sources$, returns a new information profile $\mathbf{f}(\vec{S})$ which represents $\vec{S}$ modified so that its entries are jointly consistent. We immediately require the following three properties of such an $\mathbf{f}$:

(sc1) $\vec{S} \subseteq \mathbf{f}(\vec{S})$

(sc2) $\mathbf{f}(\vec{S})$ is consistent

(sc3) If $\vec{S}$ is consistent then $\mathbf{f}(\vec{S}) = \vec{S}$

(sc1) decrees that the modification is carried out by *weakening* the individual items of information. **(sc2)** says that the end results of all these weakenings should be jointly consistent. **(sc3)** says that if $\vec{S}$ is already consistent then no modification is necessary. In addition to these three properties, we shall also find it convenient to assume that, amongst the sources, there is one distinguished source who is *completely reliable*, in the sense that any information provided by this source can safely be assumed to be true and so should never be weakened. **We fix source n to be this completely reliable source**, and reflect this by insisting on the following rule for SC functions:

(sc4) $\mathbf{f}_n(\vec{S}) = S_n$

We will denote the set of sources *minus* n by $Sources^*$. We make the following definition:

Definition 1 *Let $\mathbf{f} : \mathcal{B}^n \to \mathcal{B}^n$ be a function. Then $\mathbf{f}$ is a* social contraction function (relative to $Sources$) *iff it satisfies* **(sc1)**–**(sc4)**.

A benefit of including **(sc4)** among our basic postulates is that it gives us access to a list of individual, "local" contraction functions — one for each $i \in Sources^*$. These functions reveal, for each source i, how any item of information from i would be weak-

ened in the face of a single second item when that item is considered completely reliable.

Definition 2 *Let* $\mathbf{f}$ *be an SC function and let* $i \in Sources^*$. *We define the function* $\ominus_i^{\mathbf{f}} : \mathcal{B} \times \mathcal{B} \rightarrow \mathcal{B}$ *by, for all* $S, T \in \mathcal{B}$, $S \ominus_i^{\mathbf{f}} T = \mathbf{f}_i(\vec{U})$, *where* $\vec{U} \in \mathcal{B}^n$ *is such that* $U_i = S, U_n = T$ *and* $U_j = \mathcal{W}$ *for all* $j \notin \{i, n\}$. *We call* $\ominus_i^{\mathbf{f}}$ i*'s individual contraction function (relative to* $\mathbf{f}$*).*

(E.g., if $n = 4$, then $S \ominus_2^{\mathbf{f}} T$ is the 2^{nd} entry of the 4-tuple $\mathbf{f}(\mathcal{W}, S, \mathcal{W}, T)$.) Thus $S \ominus_i^{\mathbf{f}} T$ represents the result — according to $\mathbf{f}$ — of weakening information S from source i so that it becomes consistent with T. The following proposition is easy to prove.

Proposition 1 *Let* $\mathbf{f}$ *be an SC function and let* $i \in Sources^*$. *Then* $\ominus_i^{\mathbf{f}}$ *satisfies* **(ic1)** $S \subseteq S \ominus_i^{\mathbf{f}} T$, **(ic2)** $(S \ominus_i^{\mathbf{f}} T) \cap T \neq \emptyset$, *and* **(ic3)** *If* $S \cap T \neq \emptyset$ *then* $S \ominus_i^{\mathbf{f}} T = S$.

The properties **(ic1)**–**(ic3)** essentially correspond to the well-known basic AGM postulates for contraction (Alchourrón et al. 1985) *minus* the Recovery postulate, which in our notation would correspond to "$S \ominus_i^{\mathbf{f}} T \subseteq S \cup T$". It will become apparent in Section 4 that the $\ominus_i^{\mathbf{f}}$ don't generally satisfy this much debated (see (Hansson 1999 pp. 71–74)) property.

Recall that a principle motivating factor behind defining SC functions was to use them as a stepping-stone to defining merging operators. From a given SC function $\mathbf{f}$, we define the merging operator $\Delta_{\mathbf{f}}$ relative to *Sources* using a kind of "generalised" Levi Identity. We set, for each information profile $\vec{S}$,

$$\Delta_{\mathbf{f}}(\vec{S}) = \bigcap_{i=1}^{n} \mathbf{f}_i(\vec{S}).$$

Our basic postulates for $\mathbf{f}$ yield a corresponding set of basic properties for $\Delta_{\mathbf{f}}$: **(sc2)** gives $\Delta_{\mathbf{f}}(\vec{S}) \neq \emptyset$, while from **(sc3)** we get that $\vec{S}$ is consistent implies $\Delta_{\mathbf{f}}(\vec{S}) = \bigcap_i S_i$. Meanwhile **(sc4)** gives us $\Delta_{\mathbf{f}}(\vec{S}) \subseteq S_n$, i.e., the result of the merging must always imply the information provided by source n. In this respect $\Delta_{\mathbf{f}}$ resembles what is referred to by Konieczny and Pino-Pérez as a *merging operator with integrity constraints*, or IC merging operators for short (Konieczny & Pino-Pérez 1999), S_n here taking the role of the integrity constraints.[1]

[1] At this point it is natural to ask whether it is possible to take the converse direction and derive an SC function *from* a given IC merging operator, just like, in belief revision, it is possible to derive a contraction operator from a given revision operator via the Harper Identity (Gärdenfors 1988). This question will be taken up in future work.

2.1 MORE POSTULATES: MINIMAL CHANGE

The postulates **(sc1)**–**(sc4)** form our core set of postulates for SC functions, but there is clearly scope for other desirable properties to be put forward. One possible source for such further postulates is the idea of *minimal change*, i.e., the idea that the modification of $\vec{S}$ to achieve consistency should be kept as "small" as possible. Our condition **(sc3)** can already be said to be a mild embodiment of this idea. In this subsection we look at a couple of ways in which it can be taken further. The first rule we consider is the following:

(sc5) For all $i \in Sources^*$, if $S_i \cap \bigcap_{j \neq i} \mathbf{f}_j(\vec{S}) \neq \emptyset$ then $\mathbf{f}_i(\vec{S}) = S_i$

The motivation behind this rule is the feeling that, for each $i \in Sources^*$, we should take $\mathbf{f}_i(\vec{S}) = S_i$ whenever possible. (Recall we already have $\mathbf{f}_n(\vec{S}) = S_n$ by **(sc4)**.) Clearly if $S_i \cap \bigcap_{j \neq i} \mathbf{f}_j(\vec{S}) \neq \emptyset$ then it is possible. It is easy to see that, in the presence of **(sc1)** and **(sc4)**, **(sc5)** implies **(sc3)**. It is also quite easy to construct simple counter-examples which show that **(sc5)** doesn't hold in general for SC functions. (E.g. define $\mathbf{f}$ by setting, for each $i \in Sources$, $\mathbf{f}_i(\vec{S}) = S_i$ if either $i = n$ or $\vec{S}$ is consistent, $\mathbf{f}_i(\vec{S}) = \mathcal{W}$ otherwise.) However, even though **(sc5)** may be appealing from a minimal change point of view, its adoption can lead to counter-intuitive results, as the following example shows:

Example 1 Suppose we have three sources, i.e., $n = 3$. Suppose source 1 provides the information $S \neq \mathcal{W}$, source 2 provides the complete opposite information $\overline{S}$, and the completely reliable source 3 provides only the trivial information $\mathcal{W}$. We first claim that for any SC function $\mathbf{f}$ relative to these sources which satisfies **(sc5)** we have either $\mathbf{f}_1(S, \overline{S}, \mathcal{W}) = S$ or $\mathbf{f}_2(S, \overline{S}, \mathcal{W}) = \overline{S}$. To see this, suppose $\mathbf{f}_1(S, \overline{S}, \mathcal{W}) \neq S$. Then, by **(sc5)**, we must have $S \cap \mathbf{f}_2(S, \overline{S}, \mathcal{W}) \cap \mathbf{f}_3(S, \overline{S}, \mathcal{W}) = \emptyset$. Now we know by **(sc4)** (or **(sc1)**) that $\mathbf{f}_3(S, \overline{S}, \mathcal{W}) = \mathcal{W}$. Hence we have $S \cap \mathbf{f}_2(S, \overline{S}, \mathcal{W}) = \emptyset$, i.e., $\mathbf{f}_2(S, \overline{S}, \mathcal{W}) \subseteq \overline{S}$. Since we also have $\overline{S} \subseteq \mathbf{f}_2(S, \overline{S}, \mathcal{W})$ by **(sc1)**, we conclude that $\mathbf{f}_2(S, \overline{S}, \mathcal{W}) = \overline{S}$ which proves the claim. Given this, we have for the corresponding merging operator that either $\Delta_{\mathbf{f}}(S, \overline{S}, \mathcal{W}) \subseteq S$ or $\Delta_{\mathbf{f}}(S, \overline{S}, \mathcal{W}) \subseteq \overline{S}$. Hence when merging S and $\overline{S}$ we are forced to accept one or the other. However one can easily imagine a situation where we are unable to find any reason to prefer S to $\overline{S}$ or vice-versa (e.g. sources 1 and 2 are equally reliable, equally convinced their information is correct etc.). In this case it would not seem irrational to with-

hold judgement on whether S or $\overline{S}$ holds in the merging and to expect, say, $\Delta_{\mathbf{f}}(S, \overline{S}, \mathcal{W}) = \mathcal{W}$. Merging using an SC function which satisfies (**sc5**) rules out this possibility.

This is reminiscent of the problems with so-called *maxichoice* contraction and revision in the belief change literature (see (Hansson 1999 pp. 76–77, 209–210)). To understand why, it is helpful to change perspective slightly. For each SC function $\mathbf{f}$ and each information profile $\vec{S}$ define the set $X_{\mathbf{f}}(\vec{S}) \subseteq Sources^*$ by

$$X_{\mathbf{f}}(\vec{S}) = \{i \in Sources^* \mid \mathbf{f}_i(\vec{S}) = S_i\}$$

In other words, given that *Sources* provides the information $\vec{S}$, $X_{\mathbf{f}}(\vec{S})$ is the set of sources (other than n) who do not weaken their information according to $\mathbf{f}$. The principle of minimal change suggests we should take $X_{\mathbf{f}}(\vec{S})$ to be an inclusion-maximal subset of $Sources^*$. This is ensured by the following rule, which bears a strong resemblance to the contraction postulate "Fullness" (Hansson 1999 p. 77) which, in turn, is a characteristic postulate of maxichoice contraction:

(**sc5+**) For all $i \in Sources^*$,
if $S_i \cap (\bigcap_{j \in X_{\mathbf{f}}(\vec{S})} S_j) \cap S_n \neq \emptyset$ then $i \in X_{\mathbf{f}}(\vec{S})$

In the presence of (**sc4**), (**sc5+**) implies (**sc5**). However, in the additional presence of the following strengthening of (**sc1**), (**sc5**) becomes *equivalent* to (**sc5+**):

(**sc1+**) For all $i \in Sources$, either $\mathbf{f}_i(\vec{S}) = S_i$
or $\mathbf{f}_i(\vec{S}) = \mathcal{W}$,

The rule (**sc1+**) says, in effect, that the information from each source is either kept or discarded completely.[2]

Although Example 1 suggests (**sc5**) may be too strong, possible weakenings of it are at hand. One, which brings the individual contraction functions into the picture, is the following:

(**sc5–**) For all $i \in Sources^*$, if $S_i \cap \bigcap_{j \neq i} \mathbf{f}_j(\vec{S}) \neq \emptyset$
then $\mathbf{f}_i(\vec{S}) \subseteq S_i \ominus_i^{\mathbf{f}} \overline{S_i}$

Note that $S_i \ominus_i^{\mathbf{f}} \overline{S_i}$ is the result of weakening S_i so that it becomes consistent with $\overline{S_i}$ and so, intuitively, contains those worlds in $\overline{S_i}$ which, at least from i's viewpoint, are considered the most plausible. Hence the consequent of (**sc5–**) essentially says that if $\mathbf{f}_i(\vec{S})$ *has* to contain worlds outside of S_i, then it should contain only the most plausible ones.

The last postulate we look at is motivated by the feeling that social contraction should be entirely expressible in terms of the individual contraction functions.

(**sc6**) For all $i \in Sources^*$,
$\mathbf{f}_i(\vec{S}) = S_i \ominus_i^{\mathbf{f}} (\bigcap_{j \neq i} \mathbf{f}_j(\vec{S}))$

This postulate can also be interpreted as saying that the outcome $\mathbf{f}(\vec{S})$ of an operation of social contraction represents a kind of *equilibrium* state. One in which each source's information S_i is weakened just enough — according to that source's own individual contraction function — to be consistent with the joint result of the weakenings of all the other sources. Since, by Proposition 1, $\ominus_i^{\mathbf{f}}$ satisfies (**ic3**), it is easy to see that any SC function satisfying (**sc6**) also satisfies (**sc5**). (In fact only the "$\subseteq$" direction of (**sc6**) is needed to prove (**sc5**).)

3 EXTENDED BELIEF NEGOTIATION MODELS

So far we have examined social contraction from a strictly postulational viewpoint. In the rest of the paper we adopt another, more procedural, perspective. In (Booth 2001) the framework of *belief negotiation models* was introduced as a framework for merging together information from just *two* different sources. The idea was that the pieces of information were weakened *incrementally* via a negotiation-like process until "common ground" was reached, i.e., until they became consistent with one another. The purpose of this section is to extend this framework so that it handles information coming from n different sources (one of which is considered completely reliable) and show how each such extended belief negotiation model $\mathcal{N}$ yields an SC function $\mathbf{f}^{\mathcal{N}}$. Let's begin with a rough description of the framework.[3]

Suppose the information profile $\vec{S}$ is provided by *Sources*. The idea is that we determine $\mathbf{f}^{\mathcal{N}}(\vec{S})$ as follows. We start off with the information profile $\vec{S}^0 = \vec{S}$. If $\vec{S}^0$ is consistent then we just take $\mathbf{f}^{\mathcal{N}}(\vec{S}) = \vec{S}^0$. But if $\vec{S}^0$ is inconsistent then we perform what may be thought of as a "round of negotiation" which is just a contest between the sources. The losers of this contest (for there may be several) must then "make some con-

[2] Precisely such an assumption is made explicitly in (Cantwell 1998). Its adoption here would effectively reduce social contraction to something akin to *belief base contraction* (Hansson 1999).

[3] We remark that this framework shares some similarities with the abstract formalisation of negotiation found in (Wooldridge & Parsons 2000). For a more detailed treatment of negotiation see (Walton & Krabbe 1995).

cessions", i.e., make some weakening of their position by admitting more possibilities, while the others stay the same. Thus we arrive at the new information profile $\vec{S}^1$ where $\vec{S}^0 \subseteq \vec{S}^1$. Now if $\vec{S}^1$ is consistent then we set $\mathbf{f}^{\mathcal{N}}(\vec{S}) = \vec{S}^1$. Otherwise the next round of negotiation takes place. Once again the losers of this round make concessions, and we keep going like this until $\vec{S}^j$ is consistent, at which point we set $\mathbf{f}^{\mathcal{N}}(\vec{S}) = \vec{S}^j$. Now let us spell this out in detail.

Let Ω denote the set of all finite sequences of information profiles. Given $\omega = (\vec{S}^0, \ldots, \vec{S}^m) \in \Omega$ we will say ω is *increasing* iff $\vec{S}^j \subseteq \vec{S}^{j+1}$ for all $j = 0, 1, \ldots, m-1$. We define the set of sequences $\Sigma \subseteq \Omega$ by

$$\Sigma = \{\omega = (\vec{S}^0, \ldots, \vec{S}^m) \in \Omega \mid \omega \text{ is increasing, and } \vec{S}^m \text{ is inconsistent}\}.$$

A sequence $\sigma = (\vec{S}^0, \ldots, \vec{S}^m) \in \Sigma$ represents a possible stage in the unfinished (since $\vec{S}^m$ is inconsistent) negotiation process starting with $\vec{S}^0$. Here, the information profile $\vec{S}^m$ describes the *current standpoints* of the sources at stage σ. Given $j < m$, we let σ_j denote that sequence consisting of the first $j+1$ entries in σ, i.e., $\sigma_j = (\vec{S}^0, \ldots, \vec{S}^j)$.

In the simple negotiation scenario described above there were two ingredients in need of further specification. Firstly, we need to know how a round of negotiation is carried out. To begin with, we don't worry about the precise details. We simply assume the existence of a function $g : \Sigma \to 2^{Sources^*}$ which selects, at each negotiation stage σ, which parties should make concessions. In other words g returns the losers of the negotiation round at stage σ. Note that here we are building in our assumption that source n is completely reliable (and so never loses a round) by taking the range of g to be $2^{Sources^*}$ rather than $2^{Sources}$. We make two more mild restrictions on g. Firstly, in order to avoid deadlock we need to assume that at least one party must weaken at each stage:

(g0a) $g(\sigma) \neq \emptyset$

Secondly, suppose we reach a negotiation stage $\sigma = (\vec{S}^0, \ldots, \vec{S}^m)$ such that $S_i^m = \mathcal{W}$ for some $i \in Sources^*$. Then obviously at this stage i's information cannot be weakened any further. We restrict g so that it selects only sources who still have "room to manoeuvre".

(g0b) $i \in g(\sigma)$ implies $S_i^m \neq \mathcal{W}$ (where $\sigma = (\vec{S}^0, \ldots, \vec{S}^m)$)

The second missing ingredient is then to decide what concessions the losers of a negotiation round should make. Once again we initially abstract away from the actual process used to determine this and assume only that we are given, for each $\sigma = (\vec{S}^0, \ldots, \vec{S}^m) \in \Sigma$, a function $\blacktriangledown_\sigma : Sources^* \to \mathcal{B}$ with the interpretation that $\blacktriangledown_\sigma(i)$ represents the weakening of S_i^m given that i must weaken at stage σ. Once again to avoid deadlock, we require that this weakening be strict, unless of course $S_i^m = \mathcal{W}$:

(▼0a) $S_i^m \subseteq \blacktriangledown_\sigma(i)$

(▼0b) $\blacktriangledown_\sigma(i) = S_i^m$ implies $S_i^m = \mathcal{W}$

We can now make the following definition.

Definition 3 *An* extended belief negotiation model *(relative to Sources) is a pair* $\mathcal{N} = \langle g, \{\blacktriangledown_\sigma\}_{\sigma \in \Sigma}\rangle$ *where* $g : \Sigma \to 2^{Sources^*}$ *is a function which satisfies* **(g0a)** *and* **(g0b)**, *and, for each* $\sigma \in \Sigma$, $\blacktriangledown_\sigma : Sources^* \to \mathcal{B}$ *is a function which satisfies* **(▼0a)** *and* **(▼0b)**.

From now on when we write "belief negotiation model" we will mean an extended belief negotiation model in the sense of the above definition.[4]

A belief negotiation model $\mathcal{N}$ then uniquely determines, for any given information profile $\vec{S}$ provided by *Sources*, the complete process of negotiation on $\vec{S}$. This process is returned by the function $f^{\mathcal{N}} : \mathcal{B}^n \to \Omega$ given by

$$f^{\mathcal{N}}(\vec{S}) = \sigma = (\vec{S}^0, \ldots, \vec{S}^k)$$

where (i) $\vec{S}^0 = \vec{S}$, (ii) k is minimal such that $\vec{S}^k$ is consistent, and (iii) for each $0 \leq j < k$ we have, for each $i \in Sources$,

$$S_i^{j+1} = \begin{cases} \blacktriangledown_{\sigma_j}(i) & \text{if } i \in g(\sigma_j) \\ S_i^j & \text{otherwise.} \end{cases}$$

A belief negotiation model $\mathcal{N}$ thus yields a function $\mathbf{f}^{\mathcal{N}} : \mathcal{B}^n \to \mathcal{B}^n$, via $f^{\mathcal{N}}$ above, by simply taking $\mathbf{f}^{\mathcal{N}}(\vec{S}) = \vec{S}^k$. It is easy to check that $\mathbf{f}^{\mathcal{N}}$ forms an SC function, and not much harder to show that, in fact, *every* SC function can be said to arise in this way:

Theorem 1 *Let* $\mathbf{f} : \mathcal{B}^n \to \mathcal{B}^n$ *be a function. Then* $\mathbf{f}$ *is an SC function iff* $\mathbf{f} = \mathbf{f}^{\mathcal{N}}$ *for some belief negotiation model* $\mathcal{N}$.

In what follows we use $\Delta_{\mathcal{N}}$ to denote the merging operator defined from $\mathbf{f}^{\mathcal{N}}$, and $\ominus_i^{\mathcal{N}}$ to denote source i's individual contraction function $\ominus_i^{\mathbf{f}^{\mathcal{N}}}$ relative to $\mathbf{f}^{\mathcal{N}}$. A

[4] There are a couple of slight notational differences between this paper and (Booth 2001). In the latter paper the function g picked up the actual information items to be weakened rather than naming the sources from which they came. Similarly the functions $\blacktriangledown_\sigma$ were defined directly on the elements of S_i^m rather than the set of sources.

point to note about these latter functions is that they depend only on the functions $\blacktriangledown_\sigma$, i.e., we have the following result:

Proposition 2 *Let $\mathcal{N} = \langle g, \{\blacktriangledown_\sigma\}_{\sigma\in\Sigma}\rangle$ and $\mathcal{N}' = \langle g', \{\blacktriangledown_\sigma\}_{\sigma\in\Sigma}\rangle$ be two belief negotiation models which differ only on their first component. Then, for each $i \in Sources^*$, we have $\ominus_i^{\mathcal{N}} = \ominus_i^{\mathcal{N}'}$*

4 INSTANTIATING THE FRAMEWORK

A natural question to ask about the preceding framework is: where do the functions g and $\blacktriangledown_\sigma$ of a belief negotiation model come from? In this section we explore some possibilities — one for the $\blacktriangledown_\sigma$ and two for g, leading to two different concrete families of SC functions. To help us do this we first need to make some extra demands on the type of information provided by our sources. We assume that each source $i \in Sources^*$ provides not only a set $S_i \in \mathcal{B}$, but also some indication of the *plausibility* of *all* the worlds in $\mathcal{W}$. Such an indication is provided by a *ranking*.

Definition 4 *A* ranking *is a function $r : \mathcal{W} \to \mathbb{N}$. We extend such an r to a function on $2^{\mathcal{B}}$ by setting, for each $T \in \mathcal{B}$, $r(T) = \min_{w\in T} r(w)$. Given $S \in \mathcal{B}$ we say that r is a ranking* relative to *S iff $r^{-1}(0) = S$.*

Such rankings, or variants thereof, are a popular tool in knowledge representation. They can be traced back to the work of (Spohn 1988) and indeed have already been employed in the context of both merging (see e.g. (Meyer 2001, Meyer et al. 2001)) and belief revision (see e.g. (Williams 1994)). A ranking provides, for each $w \in \mathcal{W}$, a measure of the plausibility of w being the actual world. The lower $r(w)$ is, the more plausible it is considered to be. Rankings also allow us to talk about *degrees of certainty* or *belief*. Given $S \in \mathcal{B}$, we can interpret $r(\overline{S})$ as the degree of certainty that the world is in S — the higher $r(\overline{S})$ is, the more certain it is that S contains the actual world. We now assume that each time a source $i \in Sources^*$ provides the information S_i, he provides along with it a ranking relative to S_i. Formally, we assume we are given a *ranking assignment* for *Sources*:

Definition 5 *A* ranking assignment *(relative to Sources) is a function R which assigns, to each $i \in Sources^*$ and $S \in \mathcal{B}$, a ranking $[R_i(S)]$ relative to S.*

Note we assume source n does not provide a ranking, just S_n as normal.[5] Given this definition, we are now in a position to describe our first instantiation of the framework.

4.1 FIRST INSTANTIATION

How can we use a ranking assignment R to define suitable functions g and $\blacktriangledown_\sigma$? Turning first to g, our idea is this: the losers of the negotiation round at stage $\sigma = (\vec{S}^0, \ldots, \vec{S}^m)$ should be those sources i who are the *least certain* about their current standpoint S_i^m, according to the ranking which they have provided along with their initial information S_i^0. More precisely we define g_1 from R by setting

$$g_1(\sigma) = \{i \in Sources^* \mid S_i^m \neq \mathcal{W} \text{ and } [R_i(S_i^0)](\overline{S_i^m}) \text{ is minimal}\}$$

As for defining $\blacktriangledown_\sigma$, the method we choose is quite simple. We assume that, for each $\sigma = (\vec{S}^0, \ldots, \vec{S}^m) \in \Sigma$, if source i has to weaken at stage σ, he does so by adding to S_i^m those worlds not already in S_i^m which are the most plausible according to the ranking i has provided with his initial information S_i^0. More precisely we set

$$\blacktriangledown_\sigma(i) = S_i^m \cup \{w \in \overline{S_i^m} \mid [R_i(S_i^0)](w) \text{ is minimal}\}$$

Given a ranking assignment R, we let $\mathcal{N}(R)$ denote the belief negotiation model $\langle g_1, \{\blacktriangledown_\sigma\}_{\sigma\in\Sigma}\rangle$ with g_1 and the $\blacktriangledown_\sigma$ derived from R as above. (It should be clear that g_1 and the $\blacktriangledown_\sigma$ satisfy the requisite properties from Definition 3.) Let's now see an example of $\mathcal{N}(R)$ "in action".

Example 2 For this example we assume our background propositional language contains just two propositional variables, leading $\mathcal{W}$ to contain just four worlds which we denote here by a, b, c, d. We also assume that $Sources = \{1, 2, 3\}$. Suppose source 1 gives initial information $\{a\}$, source 2 gives $\{c\}$ and completely reliable source 3 gives $\mathcal{W}$ (and so effectively plays no role in the negotiation). Suppose our ranking assignment R is such that $[R_1(\{a\})]$ and $[R_2(\{c\})]$ are given as follows:

	0	1	2	3
$[R_1(\{a\})]$	a	b	c, d	
$[R_2(\{c\})]$	c		a, d	b

Here, the columns correspond to ranks. So, for example, $[R_1(\{a\})]$ gives world b a rank of 1, c a

[5] We also make an assumption of *commensurability* (Meyer et al. 2001), i.e., that all sources use the same *scale* when ranking the worlds according to plausibility.

rank of 2 etc. We construct the complete negotiation process $f^{\mathcal{N}(R)}(\{a\},\{c\},\mathcal{W}) = \sigma$ stage by stage, starting with $\sigma_0 = (\langle\{a\},\{c\},\mathcal{W}\rangle)$. Since we have obvious disagreement between sources 1 and 2, a first negotiation round is required. Now we have $[R_1(\{a\})](\overline{\{a\}}) = 1 < 2 = [R_2(\{c\})](\overline{\{c\}})$, i.e., source 1 is less certain of his current standpoint than source 2. Hence we have $g_1(\sigma_0) = \{1\}$, i.e., 1 loses the round and so must weaken. We have $\blacktriangledown_{\sigma_0}(1) = \{a\} \cup \{w \in \overline{\{a\}} \mid [R_1(\{a\})](w)$ is minimal$\}$, i.e., 1 adds to $\{a\}$ the most plausible non-a worlds according to $[R_1(\{a\})]$. Since b is the unique such world, this means $\blacktriangledown_{\sigma_0}(1) = \{a,b\}$ and so we reach the next negotiation stage $\sigma_1 = (\langle\{a\},\{c\},\mathcal{W}\rangle,\langle\{a,b\},\{c\},\mathcal{W}\rangle)$. Since consistency has still not been reached, another negotiation round is necessary. This time we have $[R_1(\{a\})](\overline{\{a,b\}}) = 2 = [R_2(\{c\})](\overline{\{c\}})$. Hence now both sources are equally certain of their current standpoints. Hence $g_1(\sigma_1) = \{1,2\}$, i.e., both sources must weaken. We have $\blacktriangledown_{\sigma_1}(1) = \{a,b\} \cup \{w \in \overline{\{a,b\}} \mid [R_1(\{a\})](w)$ is minimal$\} = \{a,b,c,d\} = \mathcal{W}$ and $\blacktriangledown_{\sigma_1}(2) = \{c\} \cup \{w \in \overline{\{c\}} \mid [R_2(\{c\})](w)$ is minimal$\} = \{a,c,d\}$. Hence we reach the next stage $\sigma_2 = (\langle\{a\},\{c\},\mathcal{W}\rangle,\langle\{a,b\},\{c\},\mathcal{W}\rangle,\langle\mathcal{W},\{a,c,d\},\mathcal{W}\rangle)$. Since we have now reached consistency, we end the process here with $f^{\mathcal{N}(R)}(\{a\},\{c\},\mathcal{W}) = \sigma_2$. From this we deduce $\mathbf{f}^{\mathcal{N}(R)}(\{a\},\{c\},\mathcal{W}) = \langle\mathcal{W},\{a,c,d\},\mathcal{W}\rangle$. For the corresponding merging operator we have $\Delta_{\mathcal{N}(R)}(\{a\},\{c\},\mathcal{W}) = \bigcap_{i=1}^{3} \mathbf{f}_i^{\mathcal{N}(R)}(\{a\},\{c\},\mathcal{W}) = \{a,c,d\}$.

As this example illustrates, the combined effect of our g_1 and the $\blacktriangledown_\sigma$ is, roughly speaking, a process in which the sources simultaneously add worlds rank by rank to their initial information until consistency is reached. In particular, this results in the following behaviour for the individual contraction functions $\ominus_i^{\mathcal{N}(R)}$:

Proposition 3 *Let R be a ranking assignment and let $i \in Sources^*$. Then, for all $S,T \in \mathcal{B}$, $S \ominus_i^{\mathcal{N}(R)} T = \{w \in \mathcal{W} \mid [R_i(S)](w) \leq [R_i(S)](T)\}$.*

From this the following can be shown:

Proposition 4 *Let R be a ranking assignment and let $i \in Sources^*$. Then the function $\ominus_i^{\mathcal{N}(R)}$ satisfies, in addition to* **(ic1)**–**(ic3)** *from Proposition 1, the following two properties:*

(ic4) $S \ominus_i^{\mathcal{N}(R)} (T_1 \cup T_2) \subseteq S \ominus_i^{\mathcal{N}(R)} T_1$

(ic5) *If* $(S \ominus_i^{\mathcal{N}(R)} (T_1 \cup T_2)) \cap T_1 \neq \emptyset$ *then* $S \ominus_i^{\mathcal{N}(R)} T_1 \subseteq S \ominus_i^{\mathcal{N}(R)} (T_1 \cup T_2)$

This means that $\ominus_i^{\mathcal{N}(R)}$ belongs to the class of contraction operators known as *severe withdrawal* operators, which were studied in (Rott & Pagnucco 1999). The rules **(ic4)** and **(ic5)** essentially correspond to the postulates ($\ddot{-}$7a) and ($\ddot{-}$8) given there. We also have the following nice characterisation of the merging operator $\Delta_{\mathcal{N}(R)}$:

Proposition 5 *Let R be a ranking assignment. Then, for all $\vec{S} \in \mathcal{B}^n$, $\Delta_{\mathcal{N}(R)}(\vec{S}) = \{w \in S_n \mid \max_{i \in Sources^*}[R_i(S_i)](w)$ is minimal$\}$.*

This "minimax" operator is a generalised version of the merging operator with integrity constraints Δ^{Max} given in (Konieczny & Pino-Pérez 1999), which employs a particular family of ranking assignments based on a notion of (symmetric) *distance* between propositional worlds. Similar operators are also discussed in (Meyer 2001, Meyer et al. 2001, Revesz 1993), and are shown to satisfy several interesting properties.

How do the SC functions $\mathbf{f}^{\mathcal{N}(R)}$ fare with regard to the minimal change postulates from Section 2.1? Well quite badly as it turns out. Indeed the ranking assignment R used in Example 2 provides a counterexample to show that the $\mathbf{f}^{\mathcal{N}(R)}$ do not, in general, satisfy even the weakest postulate **(sc5–)** mentioned there. To see this note that, in that example, we have $\{a\} \cap \mathbf{f}_2^{\mathcal{N}(R)}(\{a\},\{c\},\mathcal{W}) \cap \mathbf{f}_3^{\mathcal{N}(R)}(\{a\},\{c\},\mathcal{W}) = \{a\} \cap \{a,c,d\} \cap \mathcal{W} \neq \emptyset$. Now if $\mathbf{f}^{\mathcal{N}(R)}$ satisfied **(sc5–)** then we would conclude that $\mathbf{f}_1^{\mathcal{N}(R)}(\{a\},\{c\},\mathcal{W}) \subseteq \{a\} \ominus_1^{\mathcal{N}(R)} \overline{\{a\}}$. But $\mathbf{f}_1^{\mathcal{N}(R)}(\{a\},\{c\},\mathcal{W}) = \mathcal{W}$ and $\{a\} \ominus_1^{\mathcal{N}(R)} \overline{\{a\}} = \{a,b\}$. Hence $\mathbf{f}^{\mathcal{N}(R)}$ does not satisfy **(sc5–)**. Thus, interestingly, it seems that, while $\Delta_{\mathcal{N}(R)}$ might be quite well-behaved, there still seems to be room for improvement regarding the behaviour of $\mathbf{f}^{\mathcal{N}(R)}$.

4.2 SECOND INSTANTIATION

Our second instantiation of the framework is about taking a more *orderly* approach to the negotiation process. The idea now is that the sources in $Sources^*$ each take it in turn to weaken their information according to some given fixed running order. Each source, during his turn, repeatedly weakens his information until it becomes jointly consistent with the information of all the sources who have taken their turn already. This amounts to fixing $\mathbf{f}^{\mathcal{N}}(\vec{S})$ one element at a time, starting with $\mathbf{f}_n^{\mathcal{N}}(\vec{S}) = S_n$. So, using $\prec$ to denote a given strict total order on $Sources^*$ and assuming $i_1 \prec i_2 \prec \cdots \prec i_{n-1}$, we first focus on i_1 and repeatedly weaken S_{i_1} until it becomes consistent with S_n. The result of this weakening we will take to be $\mathbf{f}_{i_1}^{\mathcal{N}}(\vec{S})$.

Of course it may be that $S_{i_1} \cap S_n \neq \emptyset$ to begin with, in which case i_1 needn't do any weakening at all. Next we focus on i_2 and repeatedly weaken S_{i_2} until it becomes consistent with $\mathrm{f}^{\mathcal{N}}_{i_1}(\vec{S}) \cap S_n$. The result of *this* weakening we will take to be $\mathrm{f}^{\mathcal{N}}_{i_2}(\vec{S})$. Then it is the turn of i_3, and so on through the rest of the sources.

To fit this idea into our framework we need to define suitable functions g and $\blacktriangledown_\sigma$. For the former we define the function $g_2 : \Sigma \to 2^{Sources^*}$ from our given order $\prec$ by setting, for each negotiation stage $\sigma = (\vec{S}^0, \ldots, \vec{S}^m)$,

$$g_2(\sigma) = \{i\}, \text{ where } i \in Sources^* \text{ is minimal under } \prec \text{ such that } S^m_i \cap (\bigcap_{j \prec i} S^m_j) \cap S^m_n = \emptyset$$

For the $\blacktriangledown_\sigma$ we shall assume the weakenings are carried out in exactly the same manner as before with the help of a given ranking assignment R. Thus we define the belief negotiation model $\mathcal{N}(R, \prec) = \langle g_2, \{\blacktriangledown_\sigma\}_{\sigma \in \Sigma}\rangle$ where now g_2 is defined from $\prec$ as above and the $\blacktriangledown_\sigma$ are defined from R as in the previous subsection. (Again it is obvious that g_2 satisfies the requisite properties from Definition 3.) Let's give a worked example of a belief negotiation model of this type.

Example 3 Suppose once more that $\mathcal{W} = \{a, b, c, d\}$, but this time that $Sources = \{1, 2, 3, 4\}$. We suppose that our sources provide the information profile $\vec{S} = (\{d\}, \{a, b, d\}, \{c\}, \{a, b, c\})$. We will use the belief negotiation model $\mathcal{N}(R, \prec)$, where $\prec$ is such that $1 \prec 2 \prec 3$ and the ranking assignment R is such that $[R_1(\{d\})], [R_2(\{a, b, d\})]$ and $[R_3(\{c\})]$ are given as follows:

	0	1	2	3
$[R_1(\{d\})]$	d	a, b	c	
$[R_2(\{a, b, d\})]$	a, b, d		c	
$[R_3(\{c\})]$	c	d	a	b

Let's construct the sequence $f^{\mathcal{N}(R,\prec)}(\vec{S}) = \sigma$ stage by stage, starting with $\sigma_0 = (\vec{S}^0)$ where $\vec{S}^0 = \vec{S}$. Clearly $\vec{S}^0$ is inconsistent, so a first negotiation round is necessary. According to the definition of g_2, determining who must weaken at this initial negotiation stage is a matter of going through each of the sources in $Sources^*$ in the order prescribed by $\prec$ and selecting the first one for which $S^0_i \cap (\bigcap_{j \prec i} S^0_j) \cap S^0_4 = \emptyset$. Starting then with the minimal source in $Sources^*$, which is source 1, we immediately see that $S^0_1 \cap (\bigcap_{j \prec 1} S^0_j) \cap S^0_4 = S^0_1 \cap S^0_4 = \{d\} \cap \{a, b, c\} = \emptyset$. Hence source 1 is the loser of this negotiation round, i.e., $g_2(\sigma_0) = \{1\}$, and so must make some weakening. Since $\blacktriangledown_{\sigma_0}(1) = \{d\} \cup \{w \in \overline{\{d\}} \mid [R_1(\{d\})](w) \text{ is minimal}\} = \{a, b, d\}$ this leads us to the next stage $\sigma_1 = (\vec{S}^0, \vec{S}^1)$, where $\vec{S}^1 = (\{a, b, d\}, \{a, b, d\}, \{c\}, \{a, b, c\})$. Since consistency has not yet been reached, a second negotiation round is necessary. As a result of his weakening at the previous stage, source 1's current standpoint is no longer in conflict with that of source 4, i.e., we have $S^2_1 \cap S^2_4 = \{a, b, d\} \cap \{a, b, c\} \neq \emptyset$. Hence source 1 weakens no further. According to our ordering $\prec$, we must consider source 2 next. But $S^2_2 \cap (\bigcap_{j \prec 2} S^2_j) \cap S^2_4 = S^2_2 \cap S^2_1 \cap S^2_4 = \{a, b, d\} \cap \{a, b, d\} \cap \{a, b, c\} \neq \emptyset$ and so 2 need not weaken either. Since source 3 is the only source left, this means we must have $g_2(\sigma_1) = \{3\}$. Now $\blacktriangledown_{\sigma_1}(3) = \{c\} \cup \{w \in \overline{\{c\}} \mid [R_3(\{c\})](w) \text{ is minimal}\} = \{c, d\}$ which leads us to the next stage $\sigma_2 = (\vec{S}^0, \vec{S}^1, \vec{S}^2)$ where $\vec{S}^2 = (\{a, b, d\}, \{a, b, d\}, \{c, d\}, \{a, b, c\})$. Since we have still not reached consistency, source 3 is required to do yet more weakening, i.e., we have $g_2(\sigma_2) = \{3\}$. This time we have $\blacktriangledown_{\sigma_2}(3) = \{c, d\} \cup \{w \in \overline{\{c, d\}} \mid [R_3(\{c\})](w) \text{ is minimal}\} = \{a, c, d\}$ leading to the next stage $\sigma_3 = (\vec{S}^0, \vec{S}^1, \vec{S}^2, \vec{S}^3)$ where now $\vec{S}^3 = (\{a, b, d\}, \{a, b, d\}, \{a, c, d\}, \{a, b, c\})$. This time we *have* reached consistency, so the process stops here with $f^{\mathcal{N}(R,\prec)}(\vec{S}) = \sigma_3$ and $\mathrm{f}^{\mathcal{N}(R,\prec)}(\vec{S}) = \vec{S}^3 = (\{a, b, d\}, \{a, b, d\}, \{a, c, d\}, \{a, b, c\})$. For the corresponding merging operator we get $\Delta_{\mathcal{N}(R,\prec)}(\vec{S}) = \bigcap_{i=1}^4 S^3_i = \{a\}$.

Note that, by Proposition 2, the $\ominus^{\mathcal{N}(R,\prec)}_i$ are the same as the $\ominus^{\mathcal{N}(R)}_i$ from the previous subsection. Meanwhile we can characterise $\Delta_{\mathcal{N}(R,\prec)}$ with the help of the following piece of extra notation: We let $<_{lex}$ denote the lexicographic ordering on $\mathbb{N}^{n-1}$, i.e., given two tuples $\vec{x}, \vec{y} \in \mathbb{N}^{n-1}$ such that $\vec{x} = (x_1, \ldots, x_{n-1})$ and $\vec{y} = (y_1, \ldots, y_{n-1})$, we have $\vec{x} <_{lex} \vec{y}$ iff there exists j such that (i) $x_j < y_j$ and (ii) $x_i = y_i$ for all $i < j$. Then we have the following:

Proposition 6 *Let R be a ranking assignment and let $\prec$ be a strict total order on $Sources^*$. Then, assuming $i_1 \prec i_2 \prec \cdots \prec i_{n-1}$ and using r_j as an abbreviation for $[R_{i_j}(S_{i_j})]$, we have $\Delta_{\mathcal{N}(R,\prec)}(\vec{S}) = \{w \in S_n \mid (r_1(w), r_2(w), \ldots, r_{n-1}(w))$ is minimal under $<_{lex}\}$.*

Thus $\Delta_{\mathcal{N}(R,\prec)}(\vec{S})$ collects all the "best" worlds in S_n, in the special sense where one world is considered "better" than another if it is assigned lower rank by source i_1, or, in case they are assigned the same rank by i_1, it is assigned a lower rank by i_2, or, in case they are also assigned the same rank by i_2, it is assigned a lower rank by i_3, or, etc. Thus the effect when merging is that the opinion of source i is given *precedence* over that of i' whenever $i \prec i'$. Such a lexicographic approach to merging has been considered in (Meyer 2001) (see

Section 4.5 there) where the $\prec$ is interpreted as a given ordering of *reliability* on the sources, i.e., the most reliable sources are given precedence.

Finally, what can we say this time about the SC functions $\mathbf{f}^{\mathcal{N}(R,\prec)}$? First of all we may show the following:

Proposition 7 *Let R be a ranking assignment and $\prec$ a strict total order on Sources*. Then, for each information profile $\vec{S}$, we have*

$$\mathbf{f}_i^{\mathcal{N}(R,\prec)}(\vec{S}) = S_i \ominus_i^{\mathcal{N}(R,\prec)} (\mathbf{f}_n^{\mathcal{N}(R,\prec)}(\vec{S}) \cap \bigcap_{j \prec i} \mathbf{f}_j^{\mathcal{N}(R,\prec)}(\vec{S})).$$

In other words $\mathbf{f}_i^{\mathcal{N}(R,\prec)}(\vec{S})$ is equal to the result – according to i's individual contraction function relative to $\mathbf{f}^{\mathcal{N}(R,\prec)}$ – of weakening S_i to be jointly consistent with $\mathbf{f}_n^{\mathcal{N}(R,\prec)}(\vec{S})$ together with all the $\mathbf{f}_j^{\mathcal{N}(R,\prec)}(\vec{S})$ for which j precedes i according to $\prec$. Using this together with the fact that the $\ominus_i^{\mathcal{N}(R,\prec)}$ satisfy the properties **(ic4)** and **(ic5)** from Proposition 4 then allows us to prove:

Proposition 8 *Let R be a ranking assignment and $\prec$ a strict total order on Sources*. Then the SC function $\mathbf{f}^{\mathcal{N}(R,\prec)}$ satisfies* **(sc6)**.

Thus, imposing a strict "order of weakening" on the sources has forced our SC function to satisfy the "equilibrium" property **(sc6)** (and hence also **(sc5)** and **(sc5–)**).

5 CONCLUSION

We have made a start on the study of social contraction functions, which are applicable to the problem of merging information from multiple sources. The intention is that social contraction is to merging what contraction is to belief revision. We have considered both a postulational and a procedural approach, managing in the process of the latter to extend the belief negotiation framework of (Booth 2001). Our investigations are at an early stage, and much still needs to be done. From the postulational viewpoint we feel there are still many more postulates for social contraction waiting to be discovered and evaluated. From the negotiation viewpoint we looked in this paper at only two relatively simple possible ways of instantiating the basic negotiation framework. We are presently looking at various other, more complex, ways in which this can be done. One suggestion, due to Thomas Meyer, relates to the $\blacktriangledown_\sigma$-functions. Instead of blindly adding *all* the most plausible worlds not yet in source i's current standpoint S_i^m as is done in this paper, the function $\blacktriangledown_\sigma(i)$ should be more selective and add only those which are already included in at least one of the current standpoints S_j^m of the other sources at stage σ. (If none of these most plausible worlds appear in *any* of the S_j^m then $\blacktriangledown_\sigma(i)$ should add all of them as before.) Refinements such as this could lead to more interesting social contraction behaviour. Finally, we would also like to explore more fully the relationship between the merging operators derived from social contraction and the integrity constraints merging operators of (Konieczny & Pino-Pérez 1999).

Acknowledgements

This work is supported by the DFG research project "Computationale Dialektik". Thanks are due to Thomas Meyer for a couple of helpful suggestions.

References

C. Alchourrón, P. Gärdenfors and D. Makinson, On the logic of theory change: Partial meet contraction and revision functions, *Journal of Symbolic Logic* **50** (1985) 510–530.

R. Booth, A negotiation-style framework for non-prioritised revision, in *Proceedings of the Eighth Conference on Theoretical Aspects of Rationality and Knowledge (TARK 2001)* (2001) 137–150.

J. Cantwell, Resolving conflicting information, *Journal of Logic, Language and Information* **7**(2) (1998) 191–220.

P. Gärdenfors, *Knowledge in Flux*, MIT Press (1988).

S. O. Hansson, *A Textbook of Belief Dynamics*, Kluwer Academic Publishers (1999).

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)* (1998) 488–498.

S. Konieczny and R. Pino-Pérez, Merging with integrity constraints, in *Proceedings of the Fifth European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU '99)* (1999) 233–244.

I. Levi, Subjunctives, dispositions and chances, *Synthese* **34** (1977) 423–455.

P. Liberatore and M. Schaerf, Arbitration (or how to merge knowledge bases), *IEEE Transactions on Knowledge and Engineering* **10**(1) (1998) 76–90.

P. Maynard-Reid II and Y. Shoham, Belief fusion: Aggregating pedigreed belief states, *Journal of Logic,*

Language and Information, 10(2) (2001) 183–209.

T. A. Meyer, On the semantics of combination operations, *Journal of Applied Non-Classical Logics*, 11(1-2) (2001) 59–84.

T. A. Meyer, A. K. Ghose and S. Chopra, Social choice, merging and elections, in *Proceedings of the Sixth European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU 2001)* (2001) 466–477.

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)* (1993) 71–82.

H. Rott and M. Pagnucco, Severe withdrawal (and recovery), *Journal of Philosophical Logic*, 28 (1999) 501–547.

W. Spohn, Ordinal conditional functions: A dynamic theory of epistemic states, in W. L. Harper and B. Skyrms (eds.), *Causation in Decision, Belief Change and Statistics*, volume 2, (1988).

M. A. Williams, Transmutations of knowledge systems, in *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, (1994) 619–629.

D. N. Walton and E. C. W. Krabbe, *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*, Albany, State University of New York Press (1995).

M. J. Wooldridge and S. Parsons, Languages for negotiation, in *Proceedings of the Fourteenth European Conference on Artificial Intelligence (ECAI 2000)* (2000) 393–400.

Ontological Issues

Necessary Parts and Wholes in Bio-Ontologies

Stefan Schulz
Medical Informatics Department
Freiburg University Hospital
Stefan-Meier-Str. 26
D-79104 Freiburg, Germany
stschulz@uni-freiburg.de

Udo Hahn
Text Knowledge Engineering Lab
Freiburg University
Werthmannplatz 1
D-79085 Freiburg, Germany
hahn@coling.uni-freiburg.de

Abstract

Mereological relationships – relationships between parts and wholes – are essential for bio-ontologies. We emulate mereological reasoning by taxonomic reasoning based on SEP triplets, a special data structure for the encoding of part-whole relations, which is fully embedded in the formal framework of standard description logics. We extend the SEP formalism in order to account not only for the *part-of* but also for the *has-part* relation. Our focus is on the distinction between necessary wholes and necessary parts in concept definitions. We also analyze the role of the ontological primitives *count concepts*, *collections* and *mass concepts* in the life sciences domain and sketch how reasoning with these notions can be accounted for in a parsimonious knowledge representation language, using enhanced SEP structures.

1 Introduction

The large and rapidly growing body of knowledge in the life sciences domain requires foundational models that will not only aid in the interoperability of knowledge systems but also allow for sophisticated conceptual modeling and formal reasoning. A key role for ontologies of living systems will be played by the notion of 'biological structure' covering physical parts of organisms such as organs, tissues, cells, molecules. They constitute the location of biological processes, and are the targets of experimental, diagnostic and therapeutic interventions. Hence, they are crucial for the design of ontologies covering various fields in life sciences [2, 10, 11, 4].

In our previous work on knowledge representation, we already developed a formal framework for the seamless integration of partonomic and taxonomic reasoning [8] in the context of $\mathcal{ALC}$-based description logics [12], applied to human anatomy. This approach has already proved useful for the construction of a large terminological knowledge base [14], which covers anatomy and pathology concepts that were semi-automatically extracted from the UMLS [9], one of the largest medical knowledge repositories. Recently, we proposed an extension of this formal framework covering aspects of spatial reasoning as related to the topology in human anatomy [16, 15]. In this paper we will focus on mereological aspects of concept definitions in the domain of life sciences and will show how mereological reasoning can be expressed by classification-based taxonomic reasoning using a simple type of description logics.

2 Fundamental Notions in Mereology

2.1 Transitivity of PART-OF and HAS-PART

One of the most controversial issues in mereology is whether relations such as PART-OF, HAS-PART or subrelations of them can be considered as *transitive*. Linguistic and cognitive theories do not take this for granted. Cruse [5], e.g., discusses the plausibility of transitive inferences in common-sense domains drawing on several examples from anatomy: "a head has ears, the ears have lobes: ?the head has lobes?"; "an arm has a hand, the hand has fingers: ?the arm has fingers?"

Winston *et al.* [20] after thorough empirical investigations distinguish six different kinds of part-of relations and study their transitivity behavior in terms of common-sense reasoning. They, finally, conclude that part-whole relations can be considered transitive as long as "a single sense of part" is kept. This contrasts with obvious counterexamples, such as the relation of

set membership ($\in$), a subrelation of PART-OF according to [6].

In this paper, we subscribe to a formal understanding of parthood as worked out by Casati and Varzi [3], who axiomatically introduce the PART-OF relation p as the broadest possible notion of parthood and define it as being reflexive, antisymmetric and transitive. The same applies to the inverse relation HAS-PART (p^{-1}).

2.2 Symmetry between PART-OF and HAS-PART

Much of the confusion in these debates can be attributed to the lacking distinction between properties that hold for concrete individuals vs. abstract concept classes (cf. the overviev by Artale *et al.* [1]). For individual entities, it is quite trivial to state that when p holds between the individuals I_1 and I_2, then p^{-1} must hold between I_2 and I_1. The picture is getting more complex when the focus is on mereological relations between concepts instead of individuals. PART-OF and HAS-PART can – unlike IS-A – in a strict sense only link individuals but not concepts. However, ontologies dealing with the structure of living systems, such as the *University of Washington Digital Anatomist* [11] and others available by the UMLS metathesaurus, the VOXEL MAN ontology [13], or the Gene Ontology [4] make extensive use of mereological relations without ascribing a clear semantics to them. In the UMLS metathesaurus, e.g., we found that wherever PART-OF is assigned to a concept pair C_1 and C_2, this is mirrored by an inverse HAS-PART link between C_2 and C_1.

While this may be acceptable for informal semantic network-style representations with humans in the loop, once we aim at formal deductions a clear semantic characterization of the underlying regularities has to be provided. Therefore, when we stipulate a mereological relation rel_m in concept definitions within an ontology we have to clarify whether C_1 rel_m C_2 means

1. There is at least one instance of C_1 whose role rel_m is filled by an instance of C_2 (such as the POSSIBLE-PART relation used by Uschold [19]), or instead,
2. Every instance of C_1 must have the role rel_m filled by an instance of C_2.

In the following, we subscribe to the second interpretation, i.e., an existential assumption. A concept C_2, referred to in the definition of C_1 by the relation PART-OF (C_1 PART-OF C_2), is therefore a *necessary whole* of C_1. Accordingly, a concept C_2, referred to in the definition of C_1 by the relation HAS-PART (C_1 HAS-PART C_2), is therefore a *necessary part* of C_1.

This interpretation of mereological relations within ontologies corresponds to the notion of *existential dependence* [18]. Applied to the roles PART-OF and HAS-PART in concept definitions, Artale et al. [1] introduced the distinction between *essential parts* and *dependent parts*. Under these assumptions, the symmetry between PART-OF and HAS-PART is no longer justified, taking common conceptualizations of the domain into account. As an example, the concept *Cell* implies the existence of *Protein* as part. In other words, any instance of *Cell* must have an instance of *Protein* linked by the relation HAS-PART. *Protein* is therefore a *necessary part* of *Cell*. Vice versa, defining the concept *Protein*, it would be incorrect to imply that the role PART-OF be filled by *Cell*, since not every instance of *Protein* is part of a *Cell*. Similarly, every *Cell Nucleus* is part of a *Cell* — so it is correct to fill the role PART-OF by *Cell* in the definition of *Cell Nucleus*. *Cell* is therefore a necessary whole in a definitional statement of the concept *Cell Nucleus*. The contrary (every instance of *Cell* has the role HAS-PART filled by an instance of *Cell Nucleus*) is not true, since cells without nuclei exist (e.g., red blood cells).

This clearly shows that PART-OF and HAS-PART are not necessarily to be used in a symmetric way in the life sciencess domain, especially when we define general concepts such as *Protein*, *Cell*, *Nucleus*. This distinction becomes even more visible in the domain of human anatomy, where structural differences occur not only in the range of normal biological variability but also by human intervention. Hands without fingers, jaws without teeth, eyes without lenses, intestines without appendices occur as a result of surgical interventions. In anatomy, this leads to the distinction between *Canonical Anatomy*, i.e., the idealized anatomy referred to in textbooks and atlasses (where every hand has a thumb), and *Real Anatomy* (Rosse *et al.* [11] use the expression 'instantiated anatomy'), i.e., the potentially alterated structures of the real human body.

2.3 Count, Collection and Mass Concepts

Analyzing the English language preferred strings assigned to anatomical concepts in the 2001 UMLS [9] metathesaurus, we made the observation that the major part of anatomical concepts had a singular form as their preferred string (e.g. *Heart, Liver, Head, Hand, Connective Tissue, Blood*), whereas a minor part has a plural form assigned (e.g., *Cells, Leukocytes, Microvilli*). This subtle linguistic difference sheds light on the ontological distinction between *count concepts* and *mass concepts* in the first group and *collections of uni-*

form objects in the second group.[1] These findings coincide with the classification of mereologically relevant parts in terms of cardinality and compositionality by Gerstl and Pribbenow [6], who distinguish mass concepts, collections of elements and components of complex structures.

This raises the question whether one should ascribe collections an ontological status of their own. In the UMLS metathesaurus no distinction is made between collections and their elements. Parts of the source vocabularies refer to elements (singular), others to collections (plural) without reason. We claim, on the basis of our previous work on knowledge engineering for natural language processing [7], that collections should ontologically be distinguished from their elements.

According to Gerstl and Pribbenow [6], we consider the relation between a collection – defined as a uniform composition of identical objects – and each of these objects a special case of a mereological relation, *viz.* set membership (HAS-ELEMENT) [20], a subrelation of HAS-PART. Switching the focus from individuals to concepts, we postulate for the definition of a *collection concept* X_{coll} the *element* X to be a necessary part, or – more specifically – a necessary element.

Mass concepts are more cumbersome to define. The main criterion, according to Pease and Uschold (as mentioned in a contribution to the `standard-upper-ontology@ieee.org` forum) is that masses can be divided into parts, each of which is of the same type/class of thing as the whole entity was. Though this might be useful as far as commonsense reasoning is concerned with a low degree of granularity, it is hard to maintain in the biomedical domain — on the one hand, epidemiologists may refer to groups of people as instances of the mass concept *Population*, on the other hand, molecular biologists must refrain from mass concepts at all due to the atomicity of matter. Moreover, the above criterion would hold true for collection concepts, too. Masses can therefore be described as collections of objects which – in contrast to the stricter definition of collections – need not to be uniform. This latter criterion is especially relevant in the biomedical domain considering typical mass concepts (e.g., blood, tissue), but also other domains (e.g., granite, soil, wood).

[1] Most natural languages use specific markers in order to specify whether a discourse object belongs to either categories, e.g., a liver, **some** blood, **some** cells, in English; **eine** Leber, ... Blut, ... Zellen in German; **un** foie, **du** sang, **des** cellules in French.

3 Part-Whole Hierarchies: Partonomies

The original SEP triplet model. In order to address the problems discussed in the previous section we start from a representational construct which we already developed for the integration of partonomic and taxonomic reasoning [17, 8], a solution which is fully embedded into $\mathcal{ALC}$-style description logics [12]. We define the partitive relation p as the broadest possible notion of parthood and consider p axiomatically as transitive. Since $\mathcal{ALC}$ does not support the definition of transitive roles, we emulate transitive mereological reasoning by taxonomic reasoning through the introduction of the so-called **SEP triplet** formalism. Figure 1 illustrates the SEP structure for two biological concepts, *Cell* and *Cell Membrane*.

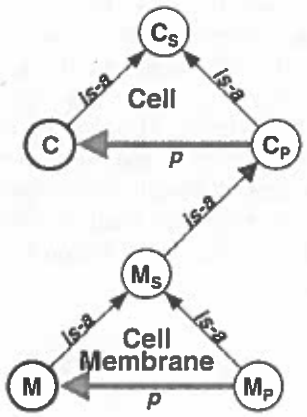


Figure 1: PART-OF Relation Modeled by SEP Triplets

An SEP triplet consists, first of all, of a composite 'structure' concept, the so-called **S-node** (here C_S, *"Cell-Structure"*, which represents cells or any part of cells). The two direct subsumees of an S-node are the corresponding **E-node** ('entity') and **P-node** ('part'), corresponding to C and C_P (*Cell* and *Cell-Part*, respectively) in Figure 1. E-nodes denote whole concepts to be modeled, whereas P-nodes are common subsumers of everything which has the role p filled by an instance of the corresponding E-nodes. Hence, for every P-node there exists a corresponding E-node for the role p. Transitivity is emulated because any subsumee of a P-node has its role p filled by an instance of the corresponding E-node. E-node and P-node can be modeled as disjoint, if this is needed. SEP triplets allow for the construction of partonomic hierarchies by subsuming S-nodes by P-nodes, such as depicted in Figure 1 where M_S (*"Cell-Membrane-Structure"*) is subsumed by C_P (*"Cell-Part"*). Parts of cell membranes (M_P) can therefore be classified as parts of cells

(C_P) — this patterns then realizes the transitivity of partitive relations.

For a formal reconstruction of these considerations, let us introduce the following TBox. We assume C and M to denote E-nodes, C_S and M_S to denote S-nodes that subsume C and M, respectively, and C_P and M_P to denote P-nodes related to C and M, respectively, via the role p. Also, since M (e.g., *Cell Membrane*) is modeled as a necessary part associated with C (e.g., *Cell*), C_P subsumes M_S (cf. Figure 1):

$$M \sqsubseteq M_S \sqsubseteq C_P \sqsubseteq C_S \sqsubseteq \mathit{BiologicalStructure} \quad (1)$$

$$C \sqsubseteq C_S \quad (2)$$

$$C_P \doteq C_S \sqcap \exists p.C \quad (3)$$

Since M is subsumed by C_P (condition 1), we infer that $\exists p.C$ holds true also for M. It is obvious that, using this pattern across various physical concepts linked with each other via the p relation, the deductions are the same — as if p were really transitive at the TBox level. 'Chains' of concepts, one having the other as filler of the role p, are modeled as P-node/S-node links between the corresponding SEP triplets. In cases where we want to obviate reflexivity we can add a disjointness criterion to condition (3) where necessary:[2]

$$C_P \doteq C_S \sqcap \neg C \sqcap \exists p.C \quad (4)$$

The SEP formalism[3] figures prominently in the design of the bio-medical ontology which forms the domain knowledge backbone of the text knowledge acquisition system MEDSYNDIKATE [7]. Within a large scalability experiment we set up a terminological knowledge base, which contains on the order of 240,000 concepts and relations [14].

The extended SEP triplet model. The SEP construct introduced up to now does not account for the inverse relation p^{-1} (HAS-PART, in the following referred to as i (INCLUDES) for better understanding). Figure 2 sketches our solution, the *extended SEP* construct. In the same way as we introduced the S-node

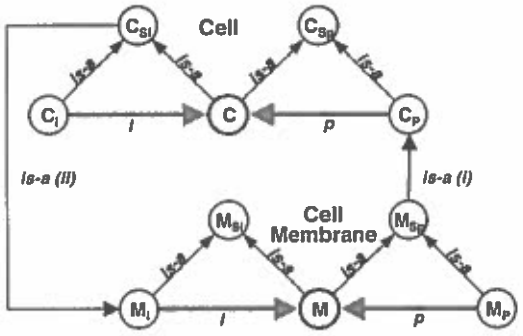


Figure 2: Extended SEP Architecture: Emulation of transitivity of both p (part-of) and i (includes-as-part) by an encoding of the concepts M and C — IS-A (I) emulates the PART-OF hierarchies, IS-A (II) HAS-PART (inclusion) hierarchies

(e.g., M_S, now relabeled as M_{Sp}) as the common subsumer of M and all of its parts, we here define M_{Si} as the common subsumer of M and all concepts that include M as part. In each extended SEP construct, the concept to be defined, here M, is then subsumed by two artificial subsumer nodes, here M_{Sp} and M_{Si}:

$$M \sqsubseteq M_{Sp} \sqcap M_{Si} \quad (5)$$

For each P-node (again defined by having the role p existentially restricted by the corresponding E-node and subsumed by the corresponding Sp-node, cf. expression 6) there is now, as its counterpart, a so-called I-(include) node which has the role i filled by the corresponding E-node and is subsumed by the corresponding Si-node (cf. expression 7).

$$M_p \doteq M_{Sp} \sqcap \exists p.M \quad (6)$$

$$M_i \doteq M_{Si} \sqcap \exists i.M \quad (7)$$

Again, I-nodes and E-nodes, on the one hand, and P-nodes and E-nodes, on the other hand can be defined as disjoint where required (cf. expressions 8 and 9).

$$M_p \doteq M_{Sp} \sqcap \neg M \sqcap \exists p.M \quad (8)$$

$$M_i \doteq M_{Si} \sqcap \neg M \sqcap \exists i.M \quad (9)$$

In the same way as we have constructed mereological hierarchies for p by cascading subsumption of Sp-nodes by P-nodes, we are now able to do the same for Si-nodes and I-nodes. In a knowledge base, this allows us to account for all direct necessary parts, necessary parts of direct parts, etc., of a given concept, simply by relying on taxonomic inheritance.

Such a knowledge base has a 3-dimensional shape. The first dimension is given by a simple taxonomy such as

[2] This is the case with countable objects which are characterized by the fact that they cannot be divided into parts so that each of the parts is the same kind of thing as the whole (following the considerations of Pease and Uschold in their contribution to the `standard-upper-ontology@ieee.org` forum).

[3] On the basis of this model, several extensions concerning topological relationships were proposed [16, 15]. These extensions are fully compatible with the enhancement described below.

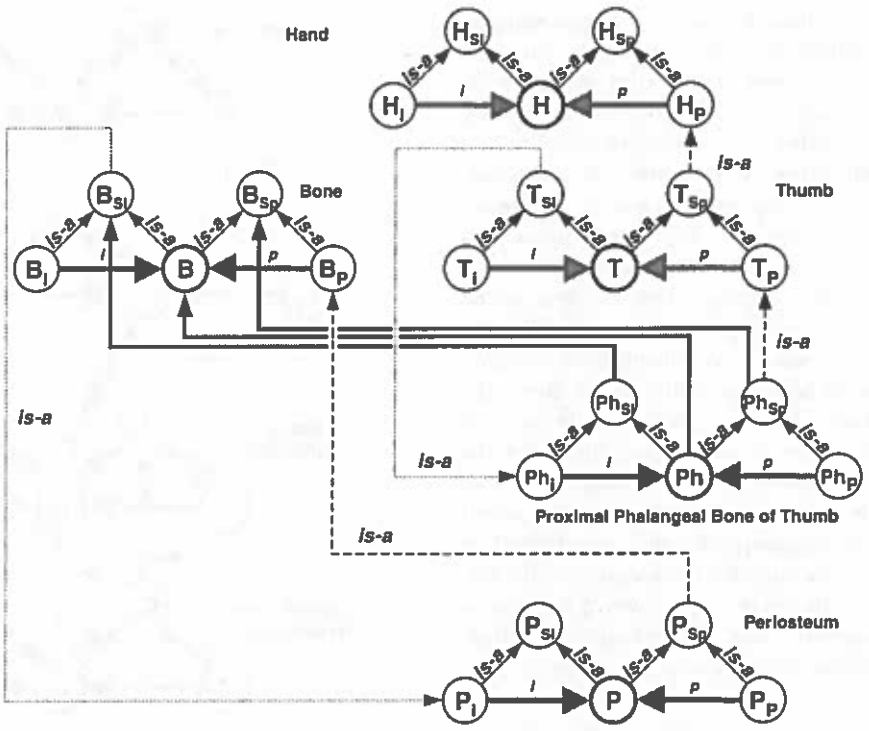


Figure 3: Extended SEP Example for Count Objects. The thick arrows depict IS-A relations between extended SEPs, the dashed arrows depict Sp-P links that emulate PART-OF relations, the dotted arrows depict Si-I links that emulate HAS-PART relations.

(*Lymphocyte* IS-A *Leukocyte* IS-A *Cell*), realized as parallel subsumption of E-nodes, Sp-nodes and Si-nodes. The second dimension is given by the PART-OF hierarchy, with *Organism* as root and taxonomic links from the Sp-node of one to the P-node of another concept. The third dimension is constituted by an "upside-down" HAS-PART hierarchy where Si-nodes are linked to I-nodes, and those concepts stand at the top which are referred to as necessary parts in concept definitions, e.g. *Cell, Protein*.[4]

In the following we will apply the extended SEP model to the three conceptual primitives *Count, Collection* and *Mass Concepts*.

Count Concepts. This is the typical situation for macroscopical objects (e.g., *head, wing, trunk*, etc.) but also for elements of collections, e.g. *cell, tooth, leaf*. Figure 3 illustrates a typical scenario from human anatomy, which instantiates the schema from Figure 2. Let us consider the concepts *Hand* (H) and *Thumb* (T), where there exists an IS-A link between T_{Sp} and H_P (for each thumb, there exists a hand it is part of), but no IS-A link between H_{Si} and T_i (not every hand needs to have a thumb). Now, we add the concept *Proximal Phalangeal Bone of the Thumb*[5] (Ph), its parent concept *Bone* (B), and the concept *Periosteum*[6] (P). Each Ph is necessarily a part of a T, as well as each T has necessarily a Ph as part. The same applies to *Bone* B and *Periosteum* P. Consequently, T_P subsumes Ph_{Sp}, as well as Ph_i subsumes T_{Si}. Ph being subsumed by B, each Ph will have a P as part (B_{Si} is subsumed by P_i), but the contrary is not the case. The model further allows the deduction that each instance of T (*Thumb*) has the role i filled by an instance of P (*Periosteum*), through the subsumption hierarchy T IS-A T_{Si} IS-A Ph_i IS-A Ph_{Si} IS-A B_{Si} IS-A P_i.

Uniform Collections. The notion of a collection of elements in a terminological knowledge base requires some clarification, since the underlying set-theoretic semantics already assigns *elements* (all individuals) to

[4]The resulting complex graph is still acyclic with respect to the IS-A relations which emulate PART-OF and HAS-PART hierarchies, given that there are no cycles in either of these subgraphs. There are no inbound links to the E-nodes, and $E \rightarrow Sp$ and $E \rightarrow Si$ are the only link types between the two subgraphs. Consequently, no path can traverse both subgraphs.

[5]The bone of the base of the thumb.

[6]The tissue which covers and nourishes a bone.

a *set* (concept class) they belong to. If the concept itself denotes a collection of elements, e.g., the concept *Leukocytes*, this means in fact that it stands for the set of all individual sets of leukocytes. Definitory statements on *Leukocytes* must therefore apply to any possible set of leukocytes. With respect to mereological inclusion, this only applies to the sets' characteristic *element* (here, *Leukocyte*). When we compare element collections with subsets of them we observe a parallelism of IS-A and PART-OF. Regarding a subset of *Leukocytes*, e.g., *Lymphocytes*, we can describe the latter in taxonomic terms — any collection of *Lymphocytes* is a collection of *Leukocytes*, but also in mereological ones — any collection of *Lymphocytes* is a subset of at least one collection of *Leukocytes*. Note that the conceptual distinction between "set concepts" and "element concepts" is necessary because reasoning about "sets of sets" is not supported by the underlying language $\mathcal{ALC}$. Concepts such as *Leukocytes*, *Cells*, etc. are considered as ordinary concepts, which exhibit, as their only characteristic, the HAS-ELEMENT relationship to their respective element concepts as their "necessary parts".

Figure 4 depicts the extended SEP encoding of a collection concept D and a subset of it, C. As an example, D may correspond to the collection concept *Leukocytes*, and C to its subset *Lymphocytes*. D subsumes C, i.e., every collection of *Lymphocytes* IS-A collection of *Leukocytes*. The link between the S-node C_{Sp} and the P-node D_P expresses that C or any part of it, e.g., a collection of *Lymphocytes*, or any part of them, is a part of a collection of *Leukocytes*.

Let us now consider the extended SEP encoding of L (*Leukocyte*), the element D is a collection of (hence the role HAS-ELEMENT of D must be filled by L). The I-node L_i subsumes all concepts that have L as part, e.g. D and C. L denotes *the* prototypical element of the collection D, in our example the concept *Leukocyte*. D has the role HAS-ELEMENT filled as an existential assumption:

$$D \sqsubseteq L_i \sqcap \exists has\text{-}element.L \qquad (10)$$

Our model allows to infer that all instances of the subset C have their role i filled by at least one instance of L, and, even more, L being subsumed by M_i, by an instance of M, as well (M, in our example, may denote the concept *Cell Membrane*). Furthermore, it becomes obvious that for many concepts there is no need for complete specifications of the complex SEP construct, since the mereological network practically melts down to a pure HAS-PART hierarchy. Thus the

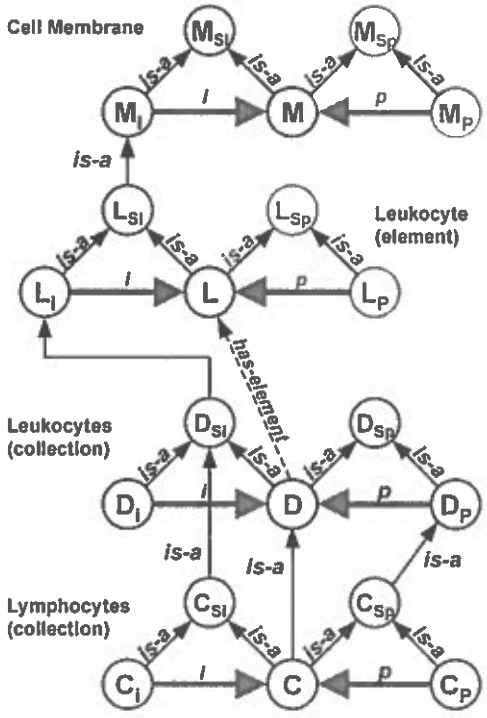


Figure 4: Sets of Elements According to the extended SEP Model

Sp- and P-nodes may be obsolete in numerous cases (grey shadowed concepts in Figure 4). Note that the denomination *uniform* collection does not contradict the fact that the collection consists of different kinds of elements. It simply refers to the level of description of the collection concept, whose entirety of elements are uniform with regard to being of the type *Leukocyte*.

Mass Concepts. The treatment of *mass concepts* is similar to that of a *collection of elements*. The extension of a mass concept such as *water* is the set of all possible portions of water in the world. The main characteristics of an instance of a mass concept is that *any portion of it has the same structure*, i.e., portioning gives rise to new instances of the same concept.[7] As already mentioned, we do not subscribe to the assumption of an atomless world, therefore mass concepts in a strict sense do not exist. Mass concepts are therefore modeled like collection concepts (Figure 4). The only difference is that the relation HAS-ELEMENT is not mandatory. It makes only sense for mass con-

[7]Here also the distinction between IS-A and PART-OF is getting increasingly fuzzy. Consider a concept such as *Blood Sample* which is still of the kind *Blood*, and has also *Blood* as a necessary whole (there must be a larger amount of blood where it is taken from).

cepts that can be described as uniform collections such as chemically pure substances, e.g., metals, cristallized proteins, oxygen, etc. Most mass concepts in living organisms however exhibit an extremely complex structure, such as *blood* or *connective tissue*. Again, we observe the predominance of HAS-PART over PART-OF. This becomes evident when we describe the components of blood, *viz.* leukocytes, thrombocytes, erythrocytes, plasma, the latter being composed by serum proteins, water, etc. None of these components is exclusively part of the blood, moreover, they can be encountered in many locations of living organisms. So we have a pure HAS-PART hierarchy which allows us to reduce the extended SEP structure to a triplet consisting of E-node, I-node and Si-node.

4 Conclusion

In this paper we proposed dedicated concept encoding patterns for mereological reasoning. Our emphasis was on expressing the transitivity of mereological relations without extending the framework of the underlying $\mathcal{ALC}$-type description logic language. The purpose was to provide an architecture by which large ontologies for the life sciences can be implemented. In order to achieve this goal, we extended the basic SEP formalism for concept encoding (in which mereological reasoning is emulated by taxonomic reasoning through the reification of the PART-OF relation) by two additional nodes in order to account for the inverse relation HAS-PART as well. The new encoding scheme, called *extended SEP* allows for the construction of part-whole hierarchies in which PART-OF and HAS-PART relations coexist independently, with support of transitive reasoning in both cases. This reasoning is, in fact, no more than taxonomic reasoning, but it produces (on a TBox level) the same results as if PART-OF and HAS-PART were indeed transitive. The price we have to pay is a considerable number of artificial concepts (increasing linearly) and, consequently, quite complicated taxonomic representation structures. This is yet a minor problem, as our experiments with very large knowledge bases have shown. On this basis, we analyzed the ontological primitives *components*, *collections of elements* and *mass concepts* and proposed templates which make use of extended SEP constructs.

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] Keith E. Campbell, A. K. Das, and Mark A. Musen. A logical foundation for representation of clinical data. *Journal of the American Medical Informatics Association*, 1(3):218–232, 1994.

[3] Roberto Casati and Achille C. Varzi. *Parts and Places. The Structures of Spatial Representation.* Cambridge, MA: MIT Press/Bradford, 1999.

[4] Gene Ontology Consortium. Creating the Gene Ontology resource: Design and implementation. *Genome Research*, 11(8):1425–1433, 2001.

[5] D. Alan Cruse. On the transitivity of the part-whole relation. *Journal of Linguistics*, 15:29–38, 1979.

[6] Peter Gerstl and Simone Pribbenow. Midwinters, end games and body parts: A classification of part-whole relations. *International Journal of Human-Computer Studies*, 43:865–889, 1995.

[7] 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.

[8] 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.

[9] NLM. *Unified Medical Language System.* Bethesda, MD: National Library of Medicine, 2001.

[10] Alan Rector, Aldo Gangemi, Galeazzi Elena, Andrzej J. Glowinski, and Angelo Rossi-Mori. The GALEN model schemata for anatomy: Towards a re-usable application-independent model of medical concepts. In P. Barahona, Veloso M., and Bryant J., editors, *MIE'94 – Medical Informatics Europe 94. Proceedings of the 12th Conference of the European Federation for Medical Informatics*, pages 229–233. Lisbon, Portugal, 1994. Amerstdam: IOS Press, 1994.

[11] Cornelius Rosse, José Leonardo V. Mejino, Bharath R. Modayur, Rex Jakobovits, Kevin P. Hinshaw, and James F. Brinkley. Motivation and

organizational principles for anatomical knowledge representation: The Digital Anatomist symbolic knowledge base. *Journal of the American Medical Informatics Association*, 5(1):17–40, 1998.

[12] Manfred Schmidt-Schauß and Gerd Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[13] Rainer Schubert, Kay Priesmeyer, Hans-Christian Wulf, and Karl-Heinz Höhne. VOXELMAN(WEB): Basistechnologie zur modellbasierten multimedialen Repräsentation von komplexen räumlichen Strukturen. *Künstliche Intelligenz*, 14(1):44–47, 2000.

[14] Stefan Schulz and Udo Hahn. Knowledge engineering by large-scale knowledge reuse: Experience from the medical domain. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *Principles of Knowledge Representation and Reasoning. Proceedings of the 7th International Conference – KR 2000*, pages 601–610. Breckenridge, CO, USA, April 12-15, 2000. San Francisco, CA: Morgan Kaufmann, 2000.

[15] Stefan Schulz and Udo Hahn. Mereotopological reasoning about parts and (w)holes in bio-ontologies. In Chris Welty and Barry Smith, editors, *Formal Ontology in Information Systems. Collected Papers from the 2nd International Conference*, pages 210–221. Ogunquit, Maine, USA, October 17-19, 2001. New York, NY: ACM Press, 2001.

[16] Stefan Schulz, Udo Hahn, and Martin Romacker. Modeling anatomical spatial relations with description logics. In J. M. Overhage, editor, *AMIA 2000 – Proceedings of the Annual Symposium of the American Medical Informatics Association. Converging Information, Technology, and Health Care*, pages 779–783. Los Angeles, CA, November 4-8, 2000. Philadelphia, PA: Hanley & Belfus, 2000.

[17] Stefan Schulz, Martin Romacker, and Udo Hahn. Part-whole reasoning in medical ontologies revisited: Introducing SEP triplets into classification-based description logics. In C. G. Chute, editor, *AMIA'98 – Proceedings of the 1998 AMIA Annual Fall Symposium. A Paradigm Shift in Health Care Information Systems: Clinical Infrastructures for the 21st Century*, pages 830–834. Orlando, FL, November 7-11, 1998. Philadelphia, PA: Hanley & Belfus, 1998.

[18] Peter Simons. *Parts: A Study in Ontology*. Oxford: Clarendon Press, 1987.

[19] Mike Uschold. The use of the typed lambda calculus for guiding naive users in the representation and acquisition of part–whole knowledge. *Data & Knowledge Engineering*, 20(3):385–404, 1996.

[20] Morton Winston, Roger Chaffin, and Douglas J. Herrmann. A taxonomy of part-whole relationships. *Cognitive Science*, 11:417–444, 1987.

Physical Objects, Identity and Vagueness

Brandon Bennett
School of Computing, University of Leeds
Leeds, LS2 9JT, UK
brandon@comp.leeds.ac.uk

Abstract

The paper presents an approach to constructing an ontology of physical objects founded on a theory of the spatio-temporal distribution of matter types. The starting point for this construction is the formal theory $\mathfrak{O}$ of 'space, time, matter and things' given in (Bennett 2001c), in which physical objects are modelled as maximal self-connected portions of some matter type. However, that theory is not compatible with the commonsense view of 'physical objects' according to which it is normal to regard an object as persisting (i.e. retaining its identity) despite the loss (or possibly gain) of small parts, which are deemed 'insignificant'. The current paper gives a more elaborate theory of the nature of physical objects and specifies their identity criteria in a way that allows for possible loss of small particles. This is achieved by explicitly taking account of an intrinsic vagueness in the identity criteria for commonsense physical objects. The paper also uses this theory to characterise various degrees of physical damage which an object can undergo.

1 Introduction

Research into representing and reasoning with spatial information has produced a number of logical languages that allow spatio-temporal situations and events to be represented at various levels of abstraction and using a variety of different conceptual vocabularies (Randell, Cui and Cohn 1992, Asher and Vieu 1995, Borgo, Guarino and Masolo 1996, Borgo, Guarino and Masolo 1997, Pratt and Schoop 1998, Bennett, Cohn, Torrini and Hazarika 2000b).

The relationship of the abstract spatial regions described by these theories to the physical objects of everyday experience is a subject that has received relatively little attention. Some formalisations do exist (e.g. (Davis 1990, Davis 1993, Shanahan 1995)) but they are surprisingly sparse in the KR literature.

A semantics which builds a theory of matter and its evolution through time onto a theory of spatial regions was given in (Bennett 2001c). However the analysis of that paper makes the assumption that physical bodies can be modelled as maximal well-connected regions of some particular matter type. But if we consider the behaviour of real world objects we soon realise that this assumption is an idealisation which cannot give a comprehensive account.

We often describe the world in terms of regular geometrical shapes. For instance, a wooden block may be described as being cubic in shape, even though when we look at it closely we find that its surface is ridged and uneven. It would be useful to have an theory which both gives an accurate ontology of the properties of real physical objects and also accounts for the value of idealising approximations in descriptions of everyday situations.

A related problem concerns the identification of an object at different times, where the object has suffered some form of damage or other alteration during the intervening period. For instance we might want to say that the same statue that was once in an ancient temple is now in a museum, even though its arms and legs are now missing. This problem becomes especially acute when we want to deal with the issue of what happens when physical objects are subject to the loss of tiny insignificant parts. Intuitively, physical objects persist despite that loss; but in order to formalise this we need to have an account of the vague notion of an 'insignificant part'. The the need for a theory of physical objects to take account of funda-

mental issues of vagueness has been recognised by a number of philosophical studies such as (Hughes 1986) and (Heller 1991). In the present paper I address these issues within a fully formal framework.

In reasoning about physical systems such a mechanical devices we often want to represent situations in which some component is broken or worn away so as to cause a malfunction. The paper concludes with a sketch of how the proposed theory of identity criteria for physical objects can be used to describe damaging events of this kind.

2 The Theory $\mathfrak{O}$ and its Idealised Objects

The analysis of this paper is conducted within the formal theory $\mathfrak{O}$, as presented in (Bennett 2001c). This is a logic specifically designed for ontology construction. I shall now briefly recap the basic syntax and semantics of $\mathfrak{O}$.

2.1 Syntax and informal semantics for $\mathfrak{O}$

$\mathfrak{O}$ contains constant symbols of five types:

- Spatial regions: $\emptyset, \ldots, r_i, \ldots$
- Time points: $\ldots, t_i, \ldots$
- Mass nouns: $\ldots, \mathbf{m}_i, \ldots$
- Individuals: $a, b, c, \ldots$
- Count nouns: $\ldots, \mathsf{C}_i, \ldots$

The region constants denote regular open subsets of the universe $\mathcal{U}$ of spatial points (which is the classical Cartesian space — $\mathcal{U} = \Re^3$) and $\emptyset$ denotes the 'empty region' (i.e. the null set of points).

Time is modelled as a set of 'histories' each of which is a mapping from time points to *states* of the universe. The semantic interpretation function of $\mathfrak{O}$ evaluates propositions relative to some history and time point (which are regarded as the 'actual history' and 'actual time'). A history/time pair $\langle h, t \rangle$ is called an *index*.

Each state is associated with a mapping from the set of mass nouns to regular open subsets of the universe. This mapping gives the spatial extension in that state of the *matter type* referred to by each mass noun.

Each individual constant is associated with an individual function which is a mapping from each index to a regular open subset of $\mathcal{U}$. This gives the extension of that individual in that history and time point corresponding to the index. An individual may be said to *exist* at all indices for which its associated individual function takes a non-null value.

The treatment of count nouns is essentially that of Gupta (1980). At each index a count noun denotes a set of individual functions, subject to the condition that each of these functions has a non-null value at that index — i.e. they correspond to individuals which exist at that index.

The atomic propositions of $\mathfrak{O}$ take the following forms:
$\alpha = \beta, \quad \mathsf{P}(\alpha, \beta), \quad \mathsf{S}(\alpha), \quad t_1 = t_2, \quad t_1 \leq t_2, \quad \mathsf{AT}(t_1)$

Where α and β are region terms which may be: either a region variable r_i, a term of the form $\mathsf{ext}(a)$ or $\mathsf{ext}(\mathbf{m})$ or the empty region constant $\emptyset$. $\mathsf{ext}(a)$ is a logical function whose value is the spatial extension of the individual a at the index of evaluation — this is simply the value at that index of the individual function associated with a. Similarly, $\mathsf{ext}(\mathbf{m})$ gives the extension of the matter type referred to by $\mathbf{m}$ (at the index of evaluation). The theory is axiomatised so that $\mathsf{ext}(\mathbf{m})$ can only vary *continuously* with time.

P is the spatial *part* relation, S is the *sphere* predicate, $\leq$ is a (dense, linear) temporal ordering on the time points and $\mathsf{AT}(t_1)$ is true just in case t_1 denotes the 'actual time' (i.e. the time of the index at which the constant is being evaluated by the semantic interpretation function). The equality symbol asserts identity either between two region or two time terms.

As explained in (Bennett 2001c) the spatial predicates P and S are constrained to satisfy a slightly modified version of the Region-Based Geometry (**RBG**) theory (Bennett, Cohn, Torrini and Hazarika 2000c, Bennett et al. 2000b, Bennett 2001b). The modification simply adds the empty region denoted by $\emptyset$ to the domain of quantification.

As well as the usual Boolean connectives, the language of $\mathfrak{O}$ contains the following constructs. If φ and ψ are propositions then so are the following:

$\boxplus\varphi, \quad \mathsf{Holds\text{-}At}(\varphi, t), \quad \forall r[\varphi], \quad \forall t[\varphi], \quad (\forall \mathsf{C}, a)[\varphi]$

The formula $\boxplus\varphi$ asserts that φ holds for all indexes $\langle h', t \rangle$, where t is the actual time and h' is any history which coincides with the actual history for all times up to t. This means that the truth of φ is determined by the actual and past states of the universe and is independent of whatever possible future might transpire.

$\mathsf{Holds\text{-}At}(\varphi, t)$ asserts that formula φ is true at the time point denoted by t (in the actual history). The Holds-At relation allows definition of a wide range of temporal operators:

D1) $\mathsf{P}\varphi \equiv_{def}$
$(\exists t_1, t_2)[\mathsf{AT}(t_1) \wedge t_1 > t_2 \wedge \mathsf{Holds\text{-}At}(\varphi, t_2)]$

D2) $\lhd\varphi \equiv_{def} \exists t_1, t_2)[\mathsf{AT}(t_2) \wedge (t_1 < t_2) \wedge \forall t_3[(t_1 \leq t_3 < t_2) \rightarrow \mathsf{Holds\text{-}At}(\varphi, t_3)]]$

P is a standard past tense operator, $\lhd\varphi$ means that φ is 'just past' in the sense that it holds for some open time interval immediately preceding (i.e. bounded by) the present time. Defining **F** and $\rhd$, the future versions of these operators just requires reversing the directions of the inequalities between time variables.

In the current paper we shall not be greatly concerned with the structure of possible histories but we will want to say that certain things are necessarily true — i.e. true at all index points. This can be expressed by the defined $\Box$ operator:

D3) $\Box\varphi \equiv_{def} \forall t[\mathsf{Holds\text{-}At}(\circledast\forall t'[\mathsf{Holds\text{-}At}(\varphi, t')], t)]$

The quantifiers $\forall r[\varphi]$ are $\forall t[\varphi]$ range respectively over regions and time points. The sorting is indicated by the variable names. Quantification over individuals is always mediated by a count noun. In evaluating the construct $(\forall \mathsf{C}, a)[\varphi]$ at any index, the variable a ranges over all individual functions in the set denoted by C at that index.

The extension of an individual varies from index to index. Moreover it can happen that at some index two different individuals have the same extension.

D4) $a \doteq b \equiv_{def} (\mathsf{ext}(a) = \mathsf{ext}(b))$

Identity between individuals requires the that they have the same extension at all indices, so, in general individual constants a and b are inter-substitutable only if they satisfy the stronger relation $\Box(a \doteq b)$.

The semantics of individuals means that each individual is directly associated with its own (trans-world) identity criteria, as given by the corresponding individual function. The identity criteria for different types of individual are articulated by axioms characterising the meanings of count nouns. Following Gupta (1980), I define two ways in which a count noun can be predicated of an individual:

D5) $\mathsf{C}(\alpha) \equiv_{def} (\exists \mathsf{C}, x)[x \doteq \alpha]$,
D6) $\mathsf{C}[\alpha] \equiv_{def} (\exists \mathsf{C}, x)[\Box(x \doteq \alpha)]$.

The first (weaker) predication says that α has the same extension as something that is a C (for example 'That quantity of metal is a ship.'), whereas the second states that α is necessarily identical to some C and hence satisfies whatever identity criteria are common to Cs (for example 'Titanic is a ship.').

2.2 Chunks and Bodies

One of the main results of (Bennett 2001c) is the precise definition and axiomatisation of the complex count noun CHUNK-OF(**m**). The individuals satisfying this predicate correspond to maximal self-connected portions of the matter type **m**.

The semantics of chunks given in (Bennett 2001c) is based on two key assumptions; namely that:

1. chunks are maximal self-connected regions of some matter type;
2. the extensions of matter types can only change continuously.

These conditions cannot be simultaneously satisfied at a point in time where a chunk breaks into pieces or merges with another chunk. Thus, the definition of chunk is such that: when a chunk breaks into two or more parts it ceases to exist immediately before the break point and new chunks corresponding to pieces of the original chunk only come into being immediately after the break. Similarly, if two or more chunks merge together the original chunks cease to exist at the point of merger and a new chunk comes into being.

This analysis is perfectly fine as long as we realise that these 'chunks' are theoretical entities which do not directly correspond to the the objects referred to in everyday communication. When we talk of a cup or a brick we are referring to an object that continues to exist even when it is chipped or scratched. Hence we need to characterise types of object which persist through the loss of certain pieces. To construct a rigorous ontology of commonsense physical objects one needs a theory which, although it must somehow relate them to the matter from which they are formed, does not simply reduced them to idealised chunks.

To bridge the gap between these very different types of object a number of theoretical sortal types will be introduced. Specifically: *rigid sub-chunks* are parts of rigid chunks which are fixed with respect to the extension of a moving chunk; and *rigid matter portions*[1] are then defined to correspond to pieces of chunks which can persist through breaking and merging events involving the chunks in which they subsist. These concepts facilitate the definition of a generic sort of *mesoscopic rigid physical object*, characterising objects whose identity criteria are robust in the face of loss of *insignificant* pieces. Here the notion of 'insignificant piece' will characterised in terms of a *maximal*

[1] What I am calling 'rigid matter portions' correspond with what Davis (1993) calls 'chunks'; and my rigid chunks correspond roughly with his 'solid objects'.

insignificant sphere, i. Natural sortals describing types of physical object (e.g. cups or chairs) can then be defined as sub-types of mesoscopic rigid physical object satisfying appropriate qualitative predicates.

3 Rigid Matter, Breakage and Pieces

In this section I shall give a completely formal specification of the notion of a 'piece' of a chunk of rigid matter. Intuitively this may seem straightforward but in fact getting a completely precise definition is rather complex given that the matter forming a chunk may move in space and may break up or merge with other chunks of matter. We will need to specify identity criteria of pieces which enable us to (for example) trace a fragment of pottery back to an original intact cup via a series of breakages.

3.1 Rigid Matter

In the current work I focus on *rigid* objects.[2] In (Bennett 2001c) the predicate $\mathsf{Rigid}(a)$ was defined to mean that a exists only during some open time interval and all its extensions at any time point within this interval are congruent. I now define $\mathsf{RigidMatter}(\mathbf{m})$ to say that $\mathbf{m}$ is a type of rigid matter — i.e. all chunks of matter type $\mathbf{m}$ are rigid:

$$\textbf{D7)}\ \mathsf{RigidMatter}(\mathbf{m}) \equiv_{def} (\forall \textsc{chunk-of}(\mathbf{m}), c)[\mathsf{Rigid}(c)]$$

I also introduce a matter type constant $\mathbf{rm}$ of rigid matter. This constant satisfies the following axioms:

$$\textbf{A1)}\ \mathsf{RigidMatter}(\mathbf{rm})$$
$$\textbf{A2)}\ \mathsf{RigidMatter}(\mathbf{m}) \rightarrow \Box(\mathsf{P}(\mathsf{ext}(\mathbf{m}), \mathsf{ext}(\mathbf{rm})))$$

In the current work I shall assume that rigid matter is also continuous, which given its rigidity, means that it cannot be created or destroyed (though it can be smashed up and joined together).[3]

3.2 Chunk Extensions at Breakage Points

Although chunks themselves cannot exist at breakage and merger points, nevertheless, for any object that exists over an open time interval we can define its limiting extensions at each end of the interval as follows:

$$\textbf{D8)}\ \mathsf{ext}{\triangleright}(a) =_{def} \mathsf{SUM}(\lambda x[{\triangleright}\mathsf{P}(x, \mathsf{ext}(a))])$$
$$\textbf{D9)}\ \mathsf{ext}{\triangleleft}(a) =_{def} \mathsf{SUM}(\lambda x[{\triangleleft}\mathsf{P}(x, \mathsf{ext}(a))])$$

Informally, $\mathsf{ext}{\triangleright}(a)$ is defined to be the sum of those regions which are part of the extension of a during some open time interval immediately following the actual time. Wherever the extension of a is continuous, the equality $\mathsf{ext}(a) = \mathsf{ext}{\triangleright}(a) = \mathsf{ext}{\triangleleft}(a)$ must hold; but where a's extension is discontinuous, $\mathsf{ext}{\triangleright}(a)$ is the extrapolation of its the extensions in the immediate future, whereas $\mathsf{ext}{\triangleleft}(a)$ is the extrapolation of its extensions in the immediate past.

The extrapolated extension functions allow us to characterise spatial events, which happen between the destruction and creation of chunks. For example, the relation $\mathsf{BreakOff}(a, b)$, which holds at time t just in case a and b are chunks of rigid matter such that a breaks away from b at time t can be defined as follows:

$$\textbf{D10)}\ \mathsf{BreakOff}(a, b) \equiv_{def} {\triangleright}\textsc{chunk-of}(\mathbf{rm})[a] \wedge {\triangleleft}\textsc{chunk-of}(\mathbf{rm})[b] \wedge \mathsf{PP}(\mathsf{ext}{\triangleright}(a), \mathsf{ext}{\triangleleft}(b))$$

3.3 Constituent Parts of Rigid Matter

An important property of rigid matter is that the constituent parts of rigid chunks maintain a constant location relative to the boundaries of the chunk. Hence the extension of a 'rigid sub-chunk' (RSC) which is part of a rigid chunk can be traced through time by reference to the extension of the chunk within which it subsists. To fix the relative spatial relationship between the sub-chunk and its parent chunk, we use the 'congruent pairs' relation introduced in (Bennett, Cohn, Torrini and Hazarika 2000a).

$$\textbf{D11)}\ \textsc{rsc}[s] \equiv_{def} \neg(\mathsf{ext}(s) = \emptyset) \wedge (\exists \textsc{chunk-of}(\mathbf{rm}), c)(\exists r_s, r_c)[\mathsf{ext}(s) = r_s \wedge \mathsf{ext}(c) = r_c \wedge \mathsf{P}(r_s, r_c) \wedge \Box((\mathsf{ext}(c) = \emptyset \wedge \mathsf{ext}(s) = \emptyset) \vee \mathsf{CGpairs}(r_s, r_c, \mathsf{ext}(s), \mathsf{ext}(c)))]$$

I now define the count noun RMP, standing for 'rigid matter part'. These entities are portions of rigid matter that persist through breakup events unless the actual matter portion of the RMP is broken. For example, figure 1 indicates two RMPs which subsist within a brick that is broken on a wall. After the impact Part2 becomes co-extensive with one of the chunks formed by the breakup.

Between breakup and merger events an RMP is always co-extensive with some RSC. By referring to the limiting extensions of RSCs at these events we can identify RSCs before and after such an event which share the

[2] Analysing portions of non-rigid matter (e.g. liquids) would require a rather different system of ontological concepts (see e.g. (Hayes 1985)).

[3] A more comprehensive theory would have to take account of phase changes (freezing, melting, setting etc.) during which rigidity could be acquired or lost.

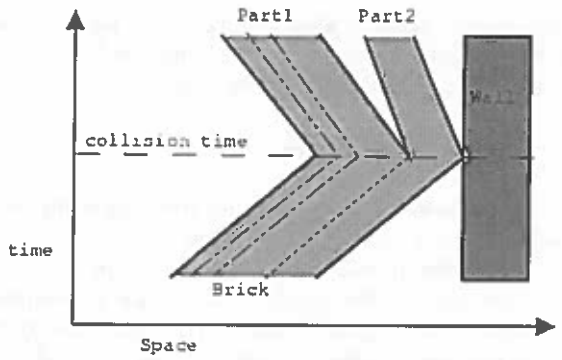


Figure 1: RMPs subsisting within a brick which is broken against a wall.

same limiting extension. These RSCs are both *component slices* of the same RMP, whose existence persists through the event. I define the more general relation of one object being a slice of another as follows:[4]

$$\textbf{D12)}\ \mathsf{Slice}(s,a) \equiv_{def} \mathsf{TCON}(s) \wedge \Box(\mathsf{ext}(s) = \emptyset \vee \mathsf{ext}(s) = \mathsf{ext}(a))$$

In order to facilitate the somewhat subtle definition of an RMP, I introduce the auxiliary relations $\mathsf{FCRSC}\triangleright(r,p)$ and $\mathsf{FCRSC}\triangleleft(r,p)$. These can be read as 'region r faces a component rigid sub-chunk of p in the immediate future/past'. In other words $\mathsf{FCRSC}\triangleright(r,p)$ and $\mathsf{FCRSC}\triangleleft(r,p)$ are true if in the immediate future/past (i.e. during some open interval following/preceding the actual time) there is an RSC which is a component slice of p and whose limiting extension at the actual time is r. Both predicates are also true if r lies within a component slice of p.

$$\textbf{D13)}\ \mathsf{FCRSC}\triangleright(r,p) \equiv_{def} (\exists^*\text{RSC}, s)[\mathsf{ext}\triangleright(s) = r \wedge \mathsf{Slice}(s,p)]$$

$$\textbf{D14)}\ \mathsf{FCRSC}\triangleleft(r,p) \equiv_{def} (\exists^*\text{RSC}, s)[\mathsf{ext}\triangleleft(s) = r \wedge \mathsf{Slice}(s,p)]$$

One will note that in the definition of FCRSC I have used a quantifier $\exists^*$ instead of the ordinary (sorted) form of quantification used in $\mathfrak{O}$. This is because, usually in that language, when one quantifies over objects of a given kind, one quantifies only over those that exist at the actual time in the actual history. In most cases this is the most useful and intuitive way to quantify; however, for the purpose of certain ontological definitions, such as that of FCRSC, it is much more convenient to quantify directly over all objects of a particular kind, whether or not they exist at the present. Hence, I define

$$\textbf{D15)}\ (\exists^*\kappa, x)[\varphi] \equiv_{def} \exists t, t'[\mathsf{AT}(t) \wedge \mathsf{Holds\text{-}At}((\exists\kappa, y)[\mathsf{Holds\text{-}At}(\varphi, t)], t')]$$

$$\textbf{D16)}\ (\forall^*\kappa, x)[\varphi] \equiv_{def} \neg(\exists^*\kappa, x)[\neg\varphi]$$

Now we come to the RMP definition. The idea is that whenever the extension of an RMP is non-empty it must face an RSC in both future and past directions. Moreover if a region faces in one direction an RSC which is a slice of an RMP, then any RSC which it faces in the other direction must also be a slice of that same RMP.

$$\begin{aligned}\textbf{D17)}\ \text{RMP}[p] \equiv_{def} {} & \neg(\mathsf{ext}(p) = \emptyset) \wedge \mathsf{TCON}(p) \wedge \\ & \Box(\exists r)[\mathsf{ext}(p) = r \wedge \\ & (r = \emptyset \vee (\mathsf{FCRSC}\triangleleft(r,p) \wedge \mathsf{FCRSC}\triangleright(r,p))) \wedge \\ & (\forall r')[(\mathsf{FCRSC}\triangleleft(r,p) \wedge (\exists^*\text{RSC}, s)[\mathsf{ext}\triangleright(s) = r']) \\ & \quad \rightarrow (r = r' \wedge \mathsf{FCRSC}\triangleright(r,p))] \wedge \\ & (\forall r')[(\mathsf{FCRSC}\triangleright(r,p) \wedge (\exists^*\text{RSC}, s)[\mathsf{ext}\triangleleft(s) = r']) \\ & \quad \rightarrow (r = r' \wedge \mathsf{FCRSC}\triangleleft(r,p))]]\end{aligned}$$

Although their definitions specify exactly what conditions an object must satisfy to be an RSC or an RMP, they do not actually guarantee the existence of any such objects. However, since these objects are supervenient on the distribution of the rigid matter in the universe (as represented by the matter type **rm**), suitable existential axioms can easily be given:

$$\textbf{A3)}\ (\forall\text{CHUNK-OF}(\mathbf{rm}), c)(\forall r)[\mathsf{P}(r, \mathsf{ext}(c)) \rightarrow (\exists\text{RSC}, s)[\mathsf{ext}(s) = r]$$

$$\textbf{A4)}\ (\forall\text{RSC}, s)(\exists\text{RMP}, r)[r \doteq s]$$

3.4 Pieces

Now that we have defined RMP we can define several important relationships between chunks and *pieces* of chunks. Unfortunately there seem to be a family of rather different relations corresponding to different senses of the natural language relation 'x is a piece of y'. For instance it is debatable whether x should be taken to be already detached or merely detachable from y; or perhaps x can actually be an integral part of y. In addition, the temporal structure of the 'piece' relation is also subtle. In some contexts the term y refers to an object which exists at the same time as x, while in others it seems to refer to an object which existed prior to the loss of the piece x.

The analysis I have given is adequate to characterise any of these alternatives. To me the following definition seems natural:

$$\begin{aligned}\textbf{D18)}\ \mathsf{Piece}(a,b) \equiv_{def} {} & \text{RMP}[a] \wedge \text{CHUNK-OF}(\mathbf{rm})(a) \\ & \wedge \mathbf{P}(\text{CHUNK-OF}(\mathbf{rm})[b] \wedge \mathsf{P}(\mathsf{ext}(a), \mathsf{ext}(b)))\end{aligned}$$

[4] TCON(s) means that s exists during a connected temporal interval — see (Bennett 2001c).

Here, a is required to be a detached RMP, which was previously part of the rigid chunk b.

4 Identification and Identity of Physical Objects

In specifying a formal ontology, for each type of object we consider we are principally concerned with defining two (closely related) kinds of criterion: a) *identification* criteria, which determine the conditions under which at a particular time a given physical extension is the extension of an object of that kind; and b) *identity* criteria which enable us to say when two observations of an object of a given kind are observations of the same object.

Later I shall use the theory of chunks and pieces to define identity criteria for objects over time that allows objects to persist through events where they lose inessential pieces. However I shall first clarify a number of issues that need to be tackled.

4.1 Vagueness in Identification and Identity

The logic $\mathfrak{O}$ treats objects as having built-in precise identification and identity criteria derived from their sort. Thus, insofar as there is vagueness associated with the extension and identity criteria of an object, this must be regarded as residing within the sort by which the object is individuated, rather than independently in the object itself. For example the extension and identity criteria of a particular cup are vague just because the concept of cup is itself vague. Thus the logical form of a claim that the region r is the extension of 'this cup' can be represented by $\mathsf{S}[\mathsf{ext}(\mathsf{this}(\mathrm{CUP})) = r]$.[5] Here the S modality can be read as 'in some sense' (see (Bennett 1998)). It means that there is some *precisification* of the language under which the extension of this(CUP) is precisely r.

In most cases it is uncontroversial whether two observations of objects at different times should be said either to be observations of the same or of different thing. However, there will also be occasions where the identity of the objects is profoundly controversial. For instance if a cup were broken in such a way that a piece of the cup could still function as a makeshift cup we might or might not be inclined to say that this piece were in some sense the same cup (albeit damaged) as the original. Our willingness to ascribe this identity would depend on the particular details of the breakage and there could be many borderline cases.

[5]Here 'this' is an ostensive operator which given a sort and evaluated in a particular context (e.g. a context where a speaker is pointing in some definite direction) returns an object of that sort. If you don't like this formalisation of ostension, you could replace this(CUP) with a definite description containing the sort CUP.

4.2 Loss of Tiny Parts

One of the most difficult obstacles to formulating reasonable identity criteria for physical objects is to take account of the possibility of objects losing tiny insignificant parts. For example, if a brick is dropped or scraped with a spade, small particles of brick dust will be removed. However, everyday intuition tells us that the original brick still exists, albeit in a slightly damaged form. The big question is, how large and how many pieces can an object lose before it ceases to exist. Informally one might say something like: an object persists as long as it does not lose any 'significant' part. But of course the notion of what is 'significant' is intrinsically vague. In the following sections I shall attempt to give some formal machinery that gives us some purchase on what can be meant by a significant part. I suggest that there are actually a number of different ways in which significance can be quantified and these are in fact relevant to the wide vocabulary of natural language descriptions of objects as being 'worn', 'damaged', 'broken' etc..

4.3 Identity Depends on History

A further complication is that the question of whether an object at some time t (in some possible history) should be regarded as the same thing as some precursor object at time t' depends not only on the states of the object(s) at t and t' but also on the history of changes that occur between t' and t. For instance if I take a long stick (say 1m) and chop it into 5cm pieces, then it does not seem reasonable to take any of these pieces to be the same stick as the original. However, if I push the stick into a power tool that shaves off thin sectional slices of the stick from one end until end up with a remaining stub 5cm long, then there do seem to be grounds for saying that this stub is in some sense the same thing as the original long stick. The rational would be that the continuity of the sticks existence is preserved throughout a series of small changes in its extent but not through the sudden loss of a 'significant' part (or parts).

5 Three Modes of Significance

I have argued that identity criteria for the objects of commonsense reasoning depend essentially on certain intuitions about what parts of an object can be neglected as insignificant. Hence in this section I shall

try to formalise some of these intuitions.

The concept of a 'significant part' is extremely slippery. At least three rather different notions of significance appear to be important:

1. A part may be significant purely in virtue of its physical extension. That is it may be too small to consider important or even to small to observe.
2. A part of an object may be insignificant relative to the geometry of the extension of that object — difference between the geometry and topology of the object with and without the part is in some sense 'negligible'.
3. A part of an object may be insignificant in relation to the purpose or functionality of that object.

5.1 Absolute Measures of Size

The most obvious criterion by which a piece of matter might be judged significant is its absolute physical size. Conversely, if a piece is smaller than some particular threshold we may call it insignificant. Actually there are several ways in which we could measure the 'size' of an object including:

1. The total volume occupied by the object.
2. The maximal distance between two points in the object.
3. The diameter of the object's minimal bounding sphere.
4. The diameter of the largest sphere which is wholly within the extension of the object.

In each case a threshold of significance can be arbitrarily defined by reference to some standard sphere, which I denote by $\mathbf{i}$, which is taken to have the largest diameter of any sphere that is considered to be 'insignificant'. In other words, any sphere larger than $\mathbf{i}$ is considered to be significant. Clearly $\mathbf{i}$ satisfies:

A5) $\mathsf{S}(\mathbf{i})$

A relation, $x >_v y$, which is true when region x has a volume greater than y is 1st-order definable in **RBG**. This seems to require a rather complex definition, so in the current paper I only give a sketch of how it can be done. We note that the volume of x is greater than or equal to y iff: there is some set $R = \{\ldots r_i \ldots\}$ of non-overlapping sub-regions of x for which there is a one-to-one mapping to a set $S = \{\ldots s_i \ldots\}$ (this can contain overlapping regions), such that for each i we have $\mathsf{CG}(r_i, s_i)$ and y is a part of the sum of all the regions in S. Here, CG is the congruence relation (which is definable from P and S). Additional formal machinery is needed to specify the one-to-one mapping. One method is to restrict R and S to sets of tetrahedra each of which can be specified by four spheres centred on its corners. A set of tetrahedra can then be identified with a *scalene sum of spheres* — i.e. a region made up a sum of disjoint spheres all of different sizes (see (Borgo et al. 1996, Bennett et al. 2000a, Bennett et al. 2000c)). The required mapping between the tetrahedra can then be defined in terms of the mapping between pairs of congruent spheres in two scalene sums of spheres.

Once we have the relation $>_v$, the property of a region's having a significant volume is easily defined:

D19) $\mathsf{SigVol}(x) \equiv_{def} x >_v \mathbf{i}$

If a region contains two points separated by a distance greater than the diameter of $\mathbf{i}$ it satisfies the SigLen predicate defined by:

D20) $\mathsf{SigLen}(x) \equiv_{def} \exists y \exists z[\mathsf{P}(y,x) \wedge \mathsf{P}(z,x) \wedge \forall y' \forall z'[(\mathsf{P}(y',y) \wedge \mathsf{P}(z',z)) \rightarrow \neg\exists s[\mathsf{CG}(s,\mathbf{i}) \wedge \mathsf{P}(y'+z',s)]]]$

Finally $\mathsf{SigBS}(x)$ specifies that x can only be bounded by a sphere which is 'significant'; and $\mathsf{SigIS}(x)$ says that x contains a 'significant' sphere:

D21) $\mathsf{SigBS}(x) \equiv_{def} \neg\exists s[\mathsf{CG}(s,\mathbf{i}) \wedge \mathsf{P}(x,s)]$
D22) $\mathsf{SigIS}(x) \equiv_{def} \exists s[\mathsf{CG}(s,\mathbf{i}) \wedge \mathsf{NTPP}(s,x)]$

Given that the size of the $\mathbf{i}$ is arbitrary, it may seem that this does not give use a concrete handle on what is significant. However, from the perspective of supervaluationistic logic, we can regard each choice of diameter for $\mathbf{i}$ as a precisification of a vague language containing a set of vague predicates for describing the significance of objects. Moreover, since all these concepts are defined by reference to $\mathbf{i}$, each such precisification enforces strong logical consistency requirements among the vague concepts.

Many useful concepts for characterising vague spatial relationships are definable in terms of $\mathbf{i}$. A relation of *approximate equality* between regions can be defined as follows:

D23) $(x \sim y) \equiv_{def} \neg\exists s[\mathsf{CG}(s,\mathbf{i}) \wedge ((\mathsf{P}(s,x) \wedge \mathsf{DR}(s,y)) \vee (\mathsf{P}(s,y) \wedge \mathsf{DR}(s,x)))]$

Informally this means it is not possible to insert a 'significant' region in the space between the boundaries of regions x and y. This definition has the shortcoming that hair-like extrusions or intrusions which are

too fine to contain a 'significant' region, no matter how long they are cannot make any significant difference between x and y. A better definition might be achieved by also limiting the maximum allowed distance between any part of one region and the nearest part of the other.

I also define the count noun Particle as follows:

$$\textbf{D24)}\ \mathsf{Particle}[p] \equiv_{def} \textsc{chunk-of}(\mathbf{rm})[p] \wedge \exists s[\mathsf{P}(\mathsf{ext}(p), s) \wedge \mathsf{CG}(s, \mathbf{i})]$$

So a particle is a chunk that is small enough to be contained within a sphere that is congruent to $\mathbf{i}$. Particles represent a type of insignificant object that is very useful for defining certain identity criteria and also events such as abrasion.

The concept of 'weak contact' was proposed by (Asher and Vieu 1995) to model the contact between distinct but touching physical objects. Figure 2 illustrates a weak contact seen at high magnification. In terms of $\mathbf{i}$ it is straightforward to model that kind of connection holding between objects such that their surfaces are close enough together that no significant region can be interposed between them.

Figure 2: A 'weak' contact at high magnification

5.1.1 i as a Supervaluationistic Parameter

Since the diameter of the maximal insignificant sphere $\mathbf{i}$ is not fixed relative to the geometrical primitives of $\mathfrak{O}$ it may be regarded as a parameter each of whose values corresponds to a possible precise sense of insignificant — i.e. it specifies a *precisification* of this concept. Since a range of concepts relating to spatial vagueness can be defined from this single vague primitive, one can ensure that all these concepts, while not fixed absolutely, do fit together in a coherent way. Moreover, logical inferences that follow from these definitions, independently of the actual magnitude of $\mathbf{i}$ can be seen as truly valid despite the underlying vagueness.

This means that a spatial theory including the $\mathbf{i}$ primitive functions in much the same as a *supervaluationistic* logic such as proposed by (Fine 1975) and investigated in relation to KR applications in (Bennett 1998). Given the potential association between values of $\mathbf{i}$ and *precisifications* it would be easy to enrich the logic by adding modal vagueness operators which enable one to refer to other possible values of $\mathbf{i}$. Thus, using the notation of (Bennett 1998), a formula of the form $\mathbf{S}[\Phi(\mathbf{i})]$ means that $\Phi(\mathbf{i})]$ is true 'in some sense' — i.e. for some value of $\mathbf{i}$. Use of such an operator enables one to combine without danger of introducing logical inconsistency, different pieces of information which might employ different senses of significance.

5.2 Minimal Axioms for 'Significant Part'

A second important mode in which an object might be considered significant is in terms of the relative significance of its extension in comparison with the extension of some larger region. Thus I axiomatise the relation of *significant parthood* holding between two regions.

$$\textbf{A6)}\ \mathsf{SP}(r, s) \rightarrow \mathsf{P}(r, s)$$
$$\textbf{A7)}\ \mathsf{SP}(r, r)$$
$$\textbf{A8)}\ (\mathsf{SP}(r, s) \wedge \mathsf{P}(x, s) \wedge \mathsf{P}(r, x)) \rightarrow \mathsf{SP}(x, s)$$
$$\textbf{A9)}\ (\mathsf{SP}(r, s) \wedge \mathsf{CGpairs}(r, s, r', s')) \rightarrow \mathsf{SP}(r', s')$$

A6 and **A7** simply ensure that a significant part is always a part and that every region is a significant part of itself. **A8** requires that if r is a significant part of s then any part of s which includes r is also a significant part of s. The final axiom ensures that whether r is a significant part of s depends only on the relative location of r with respect to s. It can also be argued that the SP relation should depend only on the relative size and location of the two regions involved. This can be axiomatised by requiring that the pairs r, s and r', s' are merely *similar* rather than congruent:

$$\textbf{A10)}\ (\mathsf{SP}(r, s) \wedge \mathsf{SimPairs}(r, s, r', s')) \rightarrow \mathsf{SP}(r', s')$$

5.3 Other Possible Axioms

I considered adding a further axiom

$$(\mathsf{SP}(r, s) \wedge \mathsf{P}(x, s) \wedge \mathsf{P}(r, x)) \rightarrow \mathsf{SP}(r, x)\ .$$

This claims that if r is a significant part of s then r is also a significant part of any sub-region s' of s of which it is a part (a sort of 'downward' persistence of significance'). However this is not in fact tenable, since it may be possible to find an s' which, although it contains r, does not contain some other part of s that is relevant to the significance of r. For instance figure 3 shows a region $r = a{+}b$ which is the sum of two discs (a and b) that slightly overlap. Here the region of overlap, x, is intuitively significant; but this region may not be considered a significant part of either of the discs.

This example suggests that one might want an axiom specifically guaranteeing the 'significance' of regions

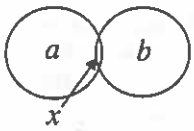


Figure 3: A significant connecting piece.

which are necessary to connect other significant parts of a region. To this end one might add:

$$\begin{aligned}\textbf{A11)}\ \exists x\exists y[&\mathsf{SP}(x, s-r) \wedge \mathsf{SP}(y, s-r) \wedge \\ &\exists c[\mathsf{ICP}(c, s) \wedge \mathsf{P}(x, c) \wedge \mathsf{P}(y, c)] \wedge \\ &\neg\exists c[\mathsf{ICP}(c, s-r) \wedge \mathsf{P}(x, c) \wedge \mathsf{P}(y, c)]] \\ &\rightarrow \mathsf{SP}(r, s)\end{aligned}$$

Where, $\mathsf{ICP}(c, s)$ is defined to mean that c is an interior-connected part of s.

Given the vagueness of SP one might wish to go further and axiomatise additional properties of different senses of the concept using a supervaluationistic logic. However for most purposes this treatment will not be required. One can argue that an appropriate degree of vagueness is captured simply by letting SP float free of precise semantics of $\mathfrak{O}$, subject to its being constrained by the above axioms. Moreover, this can be justified in terms of supervaluationistic logic on the same basis as was explained for the case of the i constant.

5.4 Identity Sufficient Parts

Finally I define two notions of significant parthood which relate RMPs to the objects in which they subsist. These complement the purely spatial significance relations defined above and enable us to describe significance in terms of higher level properties of objects.

I define the relationship $\mathsf{ISP}(p, o)$ of an RMP being an *identity sustaining piece* of an object. This relation holds whenever p is a piece of o which is sufficient to carry the identity of object o in the event of all other pieces being detached or destroyed.

$$\textbf{D25)}\ \mathsf{ISP}(p, o) \equiv_{def} \mathrm{RMP}[p] \wedge \Box(\mathsf{P}(\mathsf{ext}(p), \mathsf{ext}(o)))$$

A stronger relation of *identity necessary piece* can also be defined:

$$\begin{aligned}\textbf{D26)}\ \mathsf{INP}(p, o) \equiv_{def}\ &\mathrm{RMP}[p] \wedge \Box(\mathsf{P}(\mathsf{ext}(p), \mathsf{ext}(o)) \\ &\Box((\mathsf{ext}(p) = \emptyset \rightarrow \mathsf{ext}(o) = \emptyset))\end{aligned}$$

These relations can be used to define identity criteria for artifacts, where identity may not be reducible to questions of the significance of spatial extents, but rather is dependent on specific parts of the object.

We might want to add additional axioms constraining ISPs and/or INPs. For instance, unless all ISPs are required to have a common intersection it would be possible for an object to split into two parts each capable of sustaining its identity and this could be problematical.[6]

6 Object Types and MRPO's

We now come to that part of the ontology where we relate natural physical objects to RMPs using the various vague significance criteria. However, in order to facilitate this I introduce yet one more sort of a theoretical nature.

The vast majority of sort terms used in natural languages refer to either human artifacts, biological organisms (and organs) or geographic features. However, the identification and identity criteria of such objects are affected by a host of issues of conceptual vagueness, which obscure more even more fundamental issues.

In order to carry through my programme of establishing an ontology bottom up (starting from just the spatio-temporal distribution of matter types), I shall start by considering a generic physical object which is devoid of biological or artifactual significance. This will be called a *mesoscopic rigid physical object* (of matter type $\mathbf{m}$) and its type formally represented by the sortal $\mathrm{MRPO}(\mathbf{m})$. As the prototype of such an object may consider the sort 'rock' which denotes a portion of rigid matter with no particular structure or function. Actually 'rock' is slightly more specific than MRPO since it limits the type of rigid matter to various mineral substances and it also suggests that the material is not shaped for any purpose. The sort MRPO is neutral w.r.t. these possible limitations so that it includes rocks, lumps of wood and cups MRPO.[7]

The formal definition of MRPOs is conceptually straightforward but technically complex. The general idea is that an MRPO of matter type $\mathbf{m}$ is composed of a sequence of slices each of which is a 'significant' chunk of matter type $\mathbf{m}$. Moreover, at the time points between these slices, the difference between the limiting extensions of neighbouring component chunks are

[6] Consider a broom which has first its handle then its brush replaced. Maybe we would like both handle and brush to be ISPs? But then what if we separate these parts and join a new brush to the handle and a new handle to the brush?

[7] Consideration of biological and geographical objects would require ontological analysis which is beyond the scope of the present work. But work on formalising these domains can be found in (Cohn 2001, Bennett 2001a, Bennett 2001d).

negligible. Thus at these time points the extensions of MRPOs change discontinuously due to the gain or loss of 'insignificant' parts. (However, in so far as these parts are insignificant the change may be regarded as continuous relative to a certain commonsense abstraction of the material world.) At a point of discontinuity, the extension of the MRPO is taken to be the sum of the limiting extensions of the neighbouring chunks.[8]

I first define 'significant chunk of $\mathbf{m}$:

D27) $\text{S-CHUNK}(\mathbf{m})[x] \equiv_{def} \text{CHUNK-OF}(\mathbf{m})[x] \wedge \text{SigIS}(\text{ext}(x))$

D28) $\text{MRPO}(\mathbf{m})[p] \equiv_{def} \neg(\text{ext}(p) = \emptyset) \wedge$
$\Box((\text{ext}(p) = \emptyset) \vee$
$(\exists \text{S-CHUNK}(\mathbf{m}), c)[p \doteq c] \vee$
$(\exists^* \text{S-CHUNK}(\mathbf{m}), c_1)(\exists^* \text{S-CHUNK}(\mathbf{m}), c_2)$
$[\text{ext}_\triangleleft(c_1) \sim \text{ext}_\triangleright(c_2) \wedge$
$\text{ext}(p) = (\text{ext}_\triangleleft(c_1) + \text{ext}_\triangleright(c_2))])$
$\wedge$
$\Box((\exists^* \text{S-CHUNK}(\mathbf{m}), c)[\text{ext}_\triangleleft(p) \sim \text{ext}_\triangleright(c)] \rightarrow$
$((\text{ext}(p) = (\text{ext}_\triangleleft(p) + \text{ext}_\triangleright(c))) \wedge \triangleright(p \doteq c)))$
$\wedge$
$\Box((\exists^* \text{S-CHUNK}(\mathbf{m}), c)[\text{ext}_\triangleleft(c) \sim \text{ext}_\triangleright(p)] \rightarrow$
$((\text{ext}(p) = (\text{ext}_\triangleleft(c) + \text{ext}_\triangleright(p))) \wedge \triangleleft(p \doteq c)))$

To ensure that every significant chunk of rigid matter corresponds to a slice of some MRPO we have the following axiom:

A12) $(\text{RigidMatter}(\mathbf{m}) \wedge \text{CHUNK-OF}(\mathbf{m})[c]) \rightarrow (\exists \text{MRPO}(\mathbf{m}, x)[x \doteq c]$

6.1 Identity Criteria for Artifacts

Although, artifacts are co-extensive with purely physical bodies (which can be modelled as MRPOs), their identity and identification criteria are far more subtle.

In respect of its identification criteria, an artifact of a given kind generally has some function which it must fulfill or be capable of fulfilling in order to count as falling under that kind. For instance a CUP might be defined as 'a rigid body that has a capacity to hold liquid and (in its usual sense) can be held easily in one hand'. Containment of liquid has been addressed by (Hayes 1985) and (Randell et al. 1992).

While such properties and their roles in supplying identity criteria are undoubtedly extremely complex there seems to be no reason why they could not be articulated within $\mathfrak{O}$ in terms of relations such as ISP. Moreover, the criteria of identity through time could be specified by means of count noun definitions similar to that of MRPO but allowing for possible loss (or gain) of parts which, though noticeable (i.e. 'larger' than $\mathbf{i}$), do not result in the loss of essential properties.

Nevertheless, the specific functional properties that are necessary to a particular type of artifact will in many cases be highly controversial. Hence although we may define some formal notion of 'cup' in terms of its enduring capability to contain liquid this may not always (especially if we consider extreme examples) correspond with our intuitive (and possible inconsistent) ideas about cups. Here again, it may be possible to tackle this problem using supervaluation semantics (Fine 1975, Bennett 1998) to model the intrinsic conceptual vagueness found in the meaning of many artifact count nouns.

It is controversial whether the identity (as opposed to identification) criteria of artifacts and biological objects can be reduced the identity criteria of the MRPOs in which they subsist. It seems that this is not always the case and that to get adequate identity criteria we have to have an explicit theory of significant component parts of an artifact (such as suggested in (Simons 1987) and (Cohn 2001)).

7 Wear Damage and Breakage

I now sketch how we could apply the theory of rigid matter portions together with the various criteria of significance to the definition of a variety of kinds of destructive event which can befall a physical object.

A very important concept in the description of faults in physical mechanisms is wear cause by abrasion between two components. When one solid body rubs against another, this is likely to result in the loss of microscopic particles from one or both bodies. At first this will be unnoticeable and will not affect the working of the mechanism; however, if the abrasion continues, eventually it may result in significant distortion of a component which will cause a malfunction of the mechanism.

Abrasion can be modelled quite straightforwardly *via* the definition of particle. However, there are some issues which must be tackled. In particular there is the need to model the fact that a long series of tiny abrasions can eventually result in critical damage or breakage. This can be accommodated within the proposed theory by noting that successive loss of insignificant particles can build up to the loss of a significant part (an SP) and eventually even result in the destruction of all identity sustaining parts (ISPs).

[8] The rationale behind this is that states of contact *dominate* states of disconnection (see Galton (2000)).

Damage which is significant but does not result in the destruction of an object — e.g. the chipping or scratching of a brick — can be characterised as a loss of significant parts while retaining an identity sufficient part.

Breakage on the other hand occurs when an event results in the loss of all identity sufficient parts. As we have seen, the RMP ontology is well suited to describing breakages and enables to identify fragments as being pieces of and object which has been destroyed.

To fully characterise breakage of specific types of artifact we have to axiomatise the ISP relation in terms of whether some RMP is suitable to performing the essential functions of the artifact type. For instance a 'broken cup' might be defined as an item which once had the capacity to fulfill the function of a cup but, due to some damaging event, no-longer has this potential.

Actually there are many subtleties, so we have to be extremely careful in making such stipulations. For instance, if we break a cup we may still find that it can hold fluid within a concavity in one or more of the resulting pieces of cup. To handle this case one might say that a 'broken cup' is a (multi-piece) object made up of one or more pieces all of which were at some time parts (and indeed comprise all significant parts) of a body which was a cup, and such that for all these remaining pieces the capacity to participate in one or more of the essential functions of a cup is significantly reduced. These essential functions will include at least: capacity to hold liquid, facility to drink from, facility to hold.

8 Conclusion

It must be acknowledged that the approach pursued in the current paper is open to criticism on the grounds that its complexity renders it unsuitable for practical applications. My response is that the principal purpose of ontology is to specify the logical structure of propositions about the world in a way which is in some sense 'correct' rather than in a way that is easily manipulable by automated algorithms. Thus ontology should really only define the nature of reasoning problems rather than actually solving them.

Given that we accept this, the separate task of designing reasoning problems could still be tackled in a number of different ways. One would be to look for heuristic theorem proving algorithms that work well in many cases, despite the intractability of the problem in the worst case. Another approach (which I favour) would be to identify systems of concepts that can be reasoned with effectively (e.g. (Renz and Nebel 1999, Wolter and Zakharyaschev 2000)) and thus can be employed within a tractable sub-language of that used to articulate the whole ontology.

One might say that current paper has raised many more problems than it has actually solved. Nevertheless, I believe that I have succeeded in formalising certain concepts that are of fundamental importance in characterising the identity criteria of commonsense physical objects. In this respect the formal framework presented so far may be seen as providing a logical toolkit by means of which a more comprehensive ontological theory could be constructed. Development of such an ontology is the subject of ongoing work.

Acknowledgements

This work was supported by the EPSRC under grant GR/M56807.

Thanks to the reviewers and to Ah Lian Kor for valuable comments on the extended abstract of this paper.

References

Asher, N. and Vieu, L.: 1995, Toward a geometry of common sense: A semantics and a complete axiomatization of mereotopology, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal.*

Bennett, B.: 1998, Modal semantics for knowledge bases dealing with vague concepts, *in* A. G. Cohn, L. Schubert and S. Shapiro (eds), *Principles of Knowledge Representation and Reasoning: Proceedings of the 6th International Conference (KR-98)*, Morgan Kaufman, pp. 234–244.

Bennett, B.: 2001a, Application of supervaluation semantics to vaguely defined spatial concepts, *in* D. Montello (ed.), *Proceedings of the fifth international Conference on Spatial Information Theory (COSIT'01)*, number 2205 in *LNCS*, Springer, Morro Bay, pp. 108–123.

Bennett, B.: 2001b, A categorical axiomatisation of region-based geometry, *Fundamenta Informaticae* **46**(1–2), 145–158.

Bennett, B.: 2001c, Space, time, matter and things, *in* C. Welty and B. Smith (eds), *Proceedings of the 2nd international conference on Formal Ontology in Information Systems (FOIS'01)*, ACM, Ogunquit, pp. 105–116. `ftp://agora.leeds.ac.uk/scs/doc/srg/bennett-fois01.ps`.

Bennett, B.: 2001d, What is a forest? on the vagueness of certain geographic concepts, *Topoi* **20**(2), 189–201.

Bennett, B., Cohn, A. G., Torrini, P. and Hazarika, S. M.: 2000a, Describing rigid body motions in a qualitative theory of spatial regions, *in* H. A. Kautz and B. Porter (eds), *Proceedings of AAAI-2000*, pp. 503–509.

Bennett, B., Cohn, A. G., Torrini, P. and Hazarika, S. M.: 2000b, A foundation for region-based qualitative geometry, *in* W. Horn (ed.), *Proceedings of ECAI-2000*, pp. 204–208.

Bennett, B., Cohn, A., Torrini, P. and Hazarika, S.: 2000c, Region-based qualitative geometry, *Technical Report 2000.07*, University of Leeds, School of Computer Studies, LS2 9JT, UK. `www.scs.leeds.ac.uk/services/reports/2000.html`.

Borgo, S., Guarino, N. and Masolo, C.: 1996, A pointless theory of space based on strong congruence and connection, *in* L. C. Aiello and J. Doyle (eds), *Principles of Knowledge Representation and Reasoning, Proceedings of the 5th International Conference, KR96*, Morgan Kaufmann.

Borgo, S., Guarino, N. and Masolo, C.: 1997, An ontological theory of physical objects, *in* L. Ironi (ed.), *Proceedings of Eleventh International Workshop on Qualitative Reasoning (QR'97)*, Cortona, Italy, pp. 223–231. `http://www.ladseb.pd.cnr.it/infor/Ontology/Papers/QR97.pdf`.

Cohn, A. G.: 2001, Formalising bio-spatial knowledge, *in* C. Welty and B. Smith (eds), *Proceedings of the 2nd international conference on Formal Ontology in Information Systems (FOIS'01)*, ACM, Ogunquit.

Davis, E.: 1990, Order of magnitude reasoning in qualitative differential equations, *in* J. De Kleer and D. S. Weld (eds), *Readings in Qualitative Reasoning About Physical Systems*, Morgan Kaufmann, pp. 422–434. Revised version of technical report, Dept. Computer Science, New York University.

Davis, E.: 1993, The kinematics of cutting solid objects, *Annals of Mathematics and Artificial Intelligence* **9**(3,4), 253–305.

Fine, K.: 1975, Vagueness, truth and logic, *Synthèse* **30**, 263–300.

Galton, A. P.: 2000, Transitions in continuous time, with an application to qualitative changes in spatial relations, *in* H. B. et al. (ed.), *Advances in Temporal Logic (Proceedings of the Second International Conference on Temporal Logic, ICTL'97)*, Kluwer, pp. 279–297.

Gupta, A.: 1980, *The Logic of Common Nouns: an investigation in quantified modal logic*, Yale University Press, New Haven.

Hayes, P. J.: 1985, Naive physics I: Ontology for liquids, *in* J. R. Hobbs and B. Moore (eds), *Formal Theories of the Commonsense World*, Ablex, pp. 71–89.

Heller, M.: 1991, *The Ontology of Physical Objects*, Cambridge University Press.

Hughes, C.: 1986, Is a thing just the sum of its parts?, *Proceedings of the Aristotelian Society* **86**, 213–233.

Pratt, I. and Schoop, D.: 1998, A complete axiom system for polygonal mereotopology of the real plane, *Journal of Philosophical Logic* **27**, 621–658.

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

Renz, J. and 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.

Shanahan, M.: 1995, Default reasoning about spatial occupancy, *Artificial Intelligence*.

Simons, P.: 1987, *Parts: A Study In Ontology*, Clarendon Press, Oxford.

Wolter, F. and Zakharyaschev, M.: 2000, Spatial reasoning in RCC-8 with Boolean region terms, *Proceedings of ECAI 2000*, W. Horn, Berlin, pp. 244–248.

Uncertainty and Vagueness

A Logical Approach to Factoring Belief Networks

Adnan Darwiche
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095
darwiche@cs.ucla.edu

Abstract

We have recently proposed a tractable logical form, known as deterministic, decomposable negation normal form (d-DNNF). We have shown that d-DNNF supports a number of logical operations in polynomial time, including clausal entailment, model counting, model enumeration, model minimization, and probabilistic equivalence testing. In this paper, we discuss another major application of this logical form: the implementation of multi-linear functions (of exponential size) using arithmetic circuits (that are not necessarily exponential). Specifically, we show that each multi-linear function can be encoded using a propositional theory, and that each d-DNNF of the theory corresponds to an arithmetic circuit that implements the encoded multi-linear function. We discuss the application of these results to factoring belief networks, which can be viewed as multi-linear functions as has been shown recently. We discuss the merits of the proposed approach for factoring belief networks, and present experimental results showing how it can handle efficiently belief networks that are intractable to structure-based methods for probabilistic inference.

1 Introduction

We have recently proposed a tractable logical form, known as deterministic, decomposable negation normal form (d-DNNF) [9, 13]. We have shown that d-DNNF supports a number of logical operations in polynomial time, including clausal entailment, model counting, model enumeration, model minimization, and probabilistic equivalence testing [9, 13, 12]. In this paper, we discuss another major application of this logical form: the implementation of multi-linear functions of exponential size in terms of arithmetic circuits that are not necessarily exponential. A multi-linear function is multi-variate polynomial in which each variable has degree one—that is, the polynomial is linear in each of its variables. An arithmetic circuit is a rooted, directed acyclic graph in which leaf nodes correspond to circuit inputs, internal nodes correspond to arithmetic operations, and the root corresponds to the circuit output. One can implement a multi-linear function using an arithmetic circuit. This implementation is interesting because a multi-linear function that has an exponential size may be implemented by an arithmetic circuit whose size is not necessarily exponential. The relationship between polynomials and arithmetic circuits that implement them is a classical subject of algebraic complexity theory [30], where one is interested in the circuit complexity of certain problems—that is, the size of the smallest arithmetic circuits that can solve certain problems. Here, we will focus on the problem of generating an arithmetic circuit that implements a given multi-linear function.

The central result in this paper has two parts. First, we show that each multi-linear function can be encoded using a propositional theory. Second, we show that each d-DNNF of the theory corresponds to an arithmetic circuit which implements the encoded multi-linear function. Hence, an algorithm that converts a propositional theory into d-DNNF is immediately an algorithm for generating arithmetic circuit implementations of multi-linear functions.

We prove this result first and then discuss one of its major applications to probabilistic inference. Specifically, we have shown recently that a belief network can be represented algebraically as a multi-linear function [7, 6]. We have also shown that a large number of probabilistic queries can be obtained immediately from the partial derivatives of such a function.

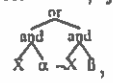
The problem however is that the multi-linear function has an exponential size, which makes it mostly of semantical interest. But if we can implement this function efficiently using an arithmetic circuit, the approach then becomes of main computational interest since the partial derivatives of an arithmetic circuit can all be computed simultaneously in time linear in the circuit size [27]. Our central result in this regard is that the multi-linear function of a belief network can be encoded efficiently using a propositional theory in Conjunctive Normal Form (CNF). Hence, by converting such a CNF into a d-DNNF of small size, we are able to compile the belief network into an arithmetic circuit, on which we can perform inference in linear time. As it turns out, the proposed approach allows us to efficiently compile some belief networks that are intractable to structure-based inference algorithms for belief networks.

This paper is structured as follows. We start by discussing deterministic, decomposable negation normal form in Section 2. We then show how multi-linear functions can be encoded using propositional theories in Section 3, and how the d-DNNF of such a theory corresponds to an arithmetic circuit implementation of the function it encodes. Section 4 is then dedicated to the application of these results to belief network inference. Experimental results are given in Section 5. We finally close with some concluding remarks in Section 6. Proofs of theorems are delegated to Appendix A.

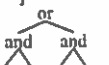
2 Deterministic, decomposable negation normal form

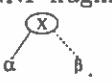
We review in this section the logical form known as deterministic, decomposable negation normal form (d-DNNF) [9, 13], and discuss its relationship to the well-known Binary Decision Diagram (BDD) [4].

A negation normal form (NNF) is a rooted, directed acyclic graph in which each leaf node is labeled with a literal, *true* or *false*, and each internal node is labeled with a conjunction $\wedge$ or disjunction $\vee$. Figure 1 depicts an example. For any node n in an NNF graph, $Vars(n)$ denotes all propositional variables that appear in the subgraph rooted at n, and $\Delta(n)$ denotes the formula represented by n and its descendants. Consider the marked node n in Figure 1(b). We have $Vars(n) = A, B$ and $\Delta(n) = (\neg A \wedge B) \vee (A \wedge \neg B)$. A number of properties can be stated on NNF graphs:

- Decomposability holds when $Vars(n_i) \cap Vars(n_j) = \emptyset$ for any two children n_i and n_j of an and-node n. This can be verified for the children of marked node in Figure 1(a).
- Determinism holds when $\Delta(n_i) \wedge \Delta(n_j)$ is logically inconsistent for any two children n_i and n_j of an or-node n. This can be verified for the children of marked node in Figure 1(b).
- Smoothness holds when $Vars(n_i) = Vars(n_j)$ for any two children n_i and n_j of an or-node n. This can be verified for the children of marked node in Figure 1(c).
- Decision holds when the root node of the NNF graph is a decision node. A *decision node* is a node labeled with *true*, *false*, or is an or-node having the form 

, where X is a variable, α and β are decision nodes. Here, X is called the *decision variable* of the node.
- Ordering is defined only for NNFs that satisfy the decision property. Ordering holds when decision variables appear in the same order along any path from the root to any leaf.

The class of decomposable negation normal form (DNNF) has been identified and studied in [8]. The class of deterministic, decomposable negation normal form (d-DNNF) has been identified and studied in [9]. Decision implies determinism. The subset of NNF that satisfies decomposability and decision (hence, determinism) corresponds to Free Binary Decision Diagrams (FBDDs) [15, 13]. The subset of NNF that satisfies decomposability, decision (hence, determinism) and ordering corresponds to Ordered Binary Decision Diagrams (OBDDs) [4, 13]. In BDD notation, however, the NNF fragment 

is drawn more compactly as 

. Hence, each internal BDD node generates three NNF nodes and six NNF edges.

Immediate from the above definitions, we have the following strict subset inclusions OBDD $\subset$ FBDD $\subset$ d-DNNF $\subset$ DNNF. Moreover, DNNF $\succ$ d-DNNF $\succ$ FBDD $\succ$ OBDD, where $\succ$ stands for "more succinct than."[1] Language L_1 is more succinct than another L_2, $L_1 \succ L_2$, iff any sentence α in L_2 has an equivalent sentence β in L_1 whose size is polynomial in the size of α; and the opposite is not true [13]. The smoothness property is not critical computationally, as it can be enforced in polynomial time [9], yet is quite helpful as we see later. The reader is referred to [13] for a comprehensive analysis of these forms and their relationships.

[1] That DNNF is strictly more succinct than d-DNNF assumes the non-collapse of the polynomial heirarchy [13].

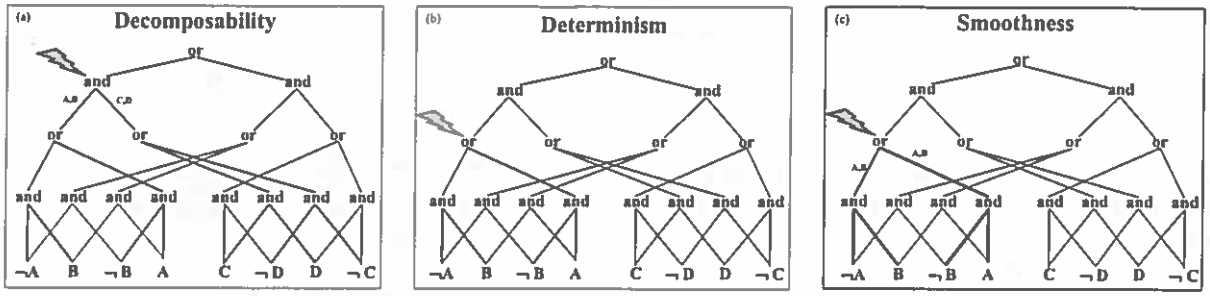


Figure 1: A negation normal form graph that satisfies decomposability, determinism and smoothness.

An algorithm for converting propositional theories in Conjunctive Normal Form (CNF) into deterministic, decomposable negation normal form (d-DNNF) is given in [10], and is used in the experimental results reported in Section 5.

3 Multi–linear functions and arithmetic circuits

We have two main goals in this section. Our first goal is to formally introduce multi–linear functions and their implementation using arithmetic circuits. Our second goal is to show how d-DNNF can be used to realize this implementation process.

A *multi–linear function* over variables Σ is a function of the form $t_1 + t_2 + \ldots t_n$, where each *term* t_i is a product of distinct variables from Σ. For example, if $\Sigma = a, b, c$, then $a+bc+ac+abc$ is a multi–linear function. There are $2^{|\Sigma|}$ distinct terms and $2^{2^{|\Sigma|}}$ distinct multi–linear functions over variables Σ.[2] Some of the multi–linear functions that come up in practice have an exponential number of terms. For example, it is shown in [7] that a belief network can be interpreted as a multi–linear function that has an exponential number of terms.

To reason with such multi–linear functions, one needs a more efficient representation. The arithmetic circuit is such a representation. An *arithmetic circuit* is a rooted, directed acyclic graph where leaves are labelled with numbers or variables, and non-leaves are labelled with arithmetic operations; see Figure 2 (right). We restrict our attention here to only addition and multiplication operations. An arithmetic circuit *implements* a multi–linear function f over variables Σ if for every value σ of these variables, the circuit will output $f(\sigma)$ under input σ. The circuit in Figure 2 implements the multi–linear function $ac + abc + c$. In the terminology of algebraic complexity theory [30], one says that the circuit in Figure 2 *computes* the function $ac + abc + c$.

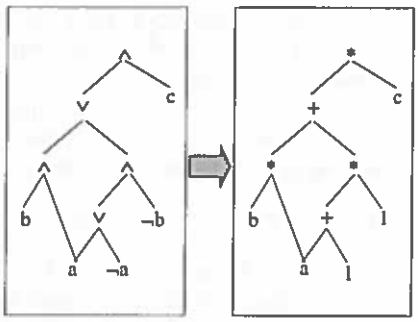


Figure 2: Extracting an arithmetic circuit from a smooth d-DNNF.

We now turn to the process of generating an arithmetic circuit representation of a given multi–linear function. Our starting point is to show that every propositional theory over variables Σ can be interpreted as encoding some multi–linear function over the same variables Σ. Consider the theory $\Delta = (a \vee \neg b) \wedge c$ over variables $\Sigma = a, b, c$ as an example. This theory has three models:

- $\sigma_1 : a = true, b = false, c = true$;
- $\sigma_2 : a = true, b = true, c = true$; and
- $\sigma_3 : a = false, b = false, c = true$.

Each one of these models σ_i can be interpreted as encoding a term t_i in the following sense. A variable appears in the term t_i iff the model σ_i sets the variable to true. That is, the first model encodes the term ac. The second model encodes the term abc, and the third model encodes the term c. The theory Δ then encodes the multi–linear function that results from adding up all these terms: $ac + abc + c$.

[2] Without loss of generality, we disallow duplicate terms in the representation of multi–linear functions.

Definition 1 *Let σ be a truth assignment over variables Σ. The term encoded by σ is defined as follows:*

$$t(\sigma) \stackrel{def}{=} \prod_{\sigma(V)=true} V.$$

Let Δ be a propositional theory over variables Σ. The multi-linear function encoded by Δ is defined as follows:

$$f(\Delta) \stackrel{def}{=} \sum_{\sigma \models \Delta} t(\sigma).$$

Propositional theories can then be used as a *specification* language for multi-linear functions. In particular, it is possible to encode a multi-linear function that has an exponential number of terms using a compact propositional theory (which has an exponential number of models). We show in Section 4 for example that the multi-linear functions corresponding to belief networks, which have an exponential number of terms, can be encoded using CNF theories of linear size.

We now have the following central result.

Definition 2 *Let Δ be a smooth d-DNNF. The arithmetic circuit encoded by Δ is obtained by replacing each conjunction in Δ by multiplication, each disjunction by addition, and each negative literal by 1.*

Figure 2 contains a smooth d-DNNF (left) and the arithmetic circuit it encodes (right).

Theorem 1 *Let Δ be a smooth d-DNNF which encodes a multi-linear function f. The arithmetic circuit encoded by Δ implements the function f.*

Therefore, if we have a multi-linear function f which is specified/encoded by a propositional theory Γ (in any form), we can generate a circuit implementation of this function by converting Γ into smooth d-DNNF. Moreover, the generated circuit has the same size as the d-DNNF. Hence, by optimizing the size of d-DNNF we are also optimizing the size of generated circuit. As mentioned earlier, we have presented a CNF to d-DNNF compiler in [10] which is quite effective on a variety of propositional theories. We use this compiler in Section 5 to compile belief networks into arithmetic circuits.

4 Factoring belief networks

We have three goals in this section. Our first goal is to review results from [7], which show that a belief network can be interpreted as a multi-linear function, and that a large number of probabilistic queries can be computed immediately using the partial derivatives of this multi-linear function. The second goal is to show that the multi-linear function of a belief network can be encoded efficiently using a CNF. Hence, a CNF to d-DNNF compiler can then be used immediately for compiling a belief network into an arithmetic circuit, which can then be used for linear time inference. Our third goal is to discuss the merits of presented approach to inference in belief networks, as compared to structure-based inference algorithms for this purpose.

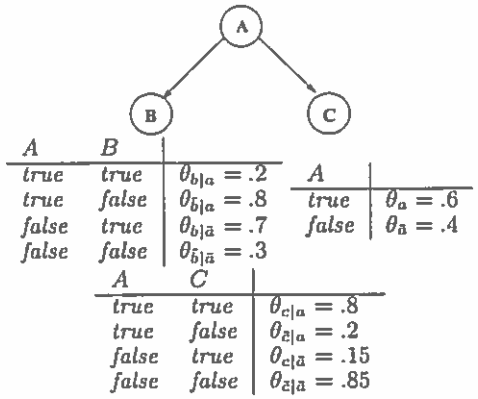


Figure 3: A belief network with its CPTs.

4.1 Belief networks as multi-linear functions

A belief network is a factored representation of a probability distribution [24]. It consists of two parts: a directed acyclic graph (DAG) and a set of conditional probability tables (CPTs). For each node X and its parents $\mathbf{U}$ in the DAG, we must have a CPT that specifies the probability distribution of X under each instantiation $\mathbf{u}$ of the parents.[3] Figure 3 depicts a simple belief network which has three CPTs.

A belief network is a "representational" factorization of a probability distribution, not a "computational" one. That is, although the network allows us to compactly represent the distribution, it needs to be processed further if one is to obtain answers to arbitrary probabilistic queries. Mainstream algorithms for inference in belief networks operate on the network to generate a "computational" factorization, allowing one to answer queries in time which is linear in the factor-

[3] We are using the standard notation: variables are denoted by upper-case letters (A) and their values by lower-case letters (a). Sets of variables are denoted by bold-face upper-case letters ($\mathbf{A}$) and their instantiations are denoted by bold-face lower-case letters ($\mathbf{a}$). For a variable A with values *true* and *false*, we use a to denote A= *true* and $\bar{a}$ to denote A= *false*. Finally, for a variable X and its parents $\mathbf{U}$, we use $\theta_{x|\mathbf{u}}$ to denote the CPT entry corresponding to $Pr(x \mid \mathbf{u})$.

ization size. One of the most influential computational factorizations of a belief network is the *jointree* [18, 17]. Standard jointree factorizations are structure-based; that is, their size depend only on the network topology and is invariant to local CPT structure. This observation has triggered much research recently for alternative, finer-grained factorizations, since real-world networks can exhibit significant local structure that can lead to significant computational savings [21, 5, 26, 31].

We discuss next one of the latest proposals in this direction, which calls for using arithmetic circuits as a computational factorization of belief networks [7, 6]. This proposal is based on viewing each belief network as a multi-linear function, which can be implemented using an arithmetic circuit. The function itself contains two types of variables:

- *Evidence indicators:* For each variable X in the network , we have a variable λ_x for each value x of X.
- *Network parameters:* For each variable X and its parents $\mathbf{U}$ in the network, we have a variable $\theta_{x|\mathbf{u}}$ for each value x of X and instantiation $\mathbf{u}$ of $\mathbf{U}$.

The multi-linear function has a term for each instantiation of the network variables, which is constructed by multiplying all evidence indicators and network parameters that are consistent with that instantiation. For example, the multi-linear function of the network in Figure 3 has 8 terms corresponding to the 8 instantiations of variables A, B, C. Three of these terms are shown below:

$$\begin{aligned} f \;=\; & \lambda_a \lambda_b \lambda_c \theta_a \theta_{b|a} \theta_{c|a} + \\ & \lambda_a \lambda_b \lambda_{\bar{c}} \theta_a \theta_{b|a} \theta_{\bar{c}|a} + \\ & \ldots \\ & \lambda_{\bar{a}} \lambda_{\bar{b}} \lambda_{\bar{c}} \theta_{\bar{a}} \theta_{\bar{b}|\bar{a}} \theta_{\bar{c}|\bar{a}}. \end{aligned}$$

Given this multi-linear function f, we can answer any query with respect to its corresponding belief network. Specifically, let $\mathbf{e}$ be an instantiation of some network variables, and suppose we want to compute the probability of $\mathbf{e}$. We can do this by simply evaluating the multi-linear function f while setting each evidence indicator λ_x to 1 if x is consistent with $\mathbf{e}$, and to 0 otherwise. For the network in Figure 3, we can compute the probability of evidence $\mathbf{e} = b\bar{c}$ by evaluating its multi-linear function above under $\lambda_a = 1, \lambda_{\bar{a}} = 1, \lambda_b = 1, \lambda_{\bar{b}} = 0$ and $\lambda_c = 0, \lambda_{\bar{c}} = 1$. This leads to $f = \theta_a \theta_{b|a} \theta_{\bar{c}|a} + \theta_{\bar{a}} \theta_{b|\bar{a}} \theta_{\bar{c}|\bar{a}}$, which equals the probability of $b, \bar{c}$ in this case. We use $f(\mathbf{e})$ to denote the result of evaluating the function f under evidence $\mathbf{e}$ as given above.

This algebraic representation of belief networks is attractive as it allows us to obtain answers to a large number of probabilistic queries directly from the derivatives of the multi-linear function. For example, the probability of any instantiation $\mathbf{e}, x$, where $X \notin \mathbf{E}$, is nothing but the partial derivative $\partial f / \partial \lambda_x$ evaluated at $\mathbf{e}$, denoted $\partial f / \partial \lambda_x(\mathbf{e})$. Moreover, the probability of any instantiation $\mathbf{e}, x, \mathbf{u}$, where $\mathbf{U}$ are the parents of X, is nothing but $\theta_{x|\mathbf{u}} \partial f / \partial \theta_{x|\mathbf{u}}(\mathbf{e})$. The ability to compute answers to probabilistic queries directly from such derivatives is valuable computationally since the (first) partial derivatives of an arithmetic circuit can all be computed simultaneously in time linear in the circuit size [7, 27]. Therefore, by implementing the multi-linear function as an arithmetic circuit, we can compute a large number of probabilistic queries in time linear the circuit size.

4.2 Encoding a belief network using a CNF

The multi-linear function of a belief network has an exponential number of terms. Yet, one can encode it efficiently using a CNF. The encoding is given next.

Definition 3 *The CNF encoding Δ of a belief network contains two types of propositional variables. For each network variable X, we have a propositional variable λ_x for each value x of X. And for each network variable X and its parents $\mathbf{U}$, we have a propositional variable $\theta_{x|\mathbf{u}}$ for each value x of X and instantiation $\mathbf{u}$ of $\mathbf{U}$. For each network variable X with values $x^1, \ldots, x^k$, the encoding Δ contains the clauses:*

$$\lambda_{x^1} \vee \ldots \vee \lambda_{x^k} \quad (1)$$

$$\neg\lambda_{x^i} \vee \neg\lambda_{x^j}, \quad i \neq j. \quad (2)$$

Moreover, for each propositional variable $\theta_{x|u_1,\ldots,u_m}$, the encoding Δ contains the clauses:

$$\lambda_x \wedge \lambda_{u_1} \wedge \ldots \wedge \lambda_{u_m} \rightarrow \theta_{x|u_1,\ldots,u_m} \quad (3)$$

$$\theta_{x|u_1,\ldots,u_m} \rightarrow \lambda_x \quad (4)$$

$$\theta_{x|u_1,\ldots,u_m} \rightarrow \lambda_{u_1}, \quad \ldots, \quad \theta_{x|u_1,\ldots,u_m} \rightarrow \lambda_{u_m}. \quad (5)$$

Clauses 1–2 say that each term of the multi-linear function must include exactly one of the evidence indicators for X. Clauses 3–5 say that a term includes evidence indicators $\lambda_x, \lambda_{u_1}, \ldots \lambda_{u_m}$ iff it includes parameter $\theta_{x|u_1,\ldots,u_m}$. Clauses 3–5 can be compactly represented as an equivalence:

$$\lambda_x \wedge \lambda_{u_1} \wedge \ldots \wedge \lambda_{u_m} \leftrightarrow \theta_{x|u_1,\ldots,u_m}. \quad (6)$$

Theorem 2 *The CNF encoding of a belief network encodes the multi-linear function of given network.*

Given that each network variable has at most k values and at most m parents, the CNF encoding contains $O(nmk^m + nk^2)$ clauses. $O(nk^2)$ of these clauses are of Types 1–2, and $O(nmk^m)$ clauses are of Types 3–5. Note that the number of CPT entries is $O(nk^m)$ in this case. The encoding suggested by Definition 3 is practical when we do not have many values (k) or parents (m) per variable. When either k or m is not small enough, a much more efficient encoding would result if one uses a constraint–based language which allows multi-valued variables. We do not purse this approach here though, as it would require a d-DNNF compiler which can handle multi–valued variables. The d-DNNF compiler we used from [10] does not handle such variables yet.

Let us now consider an example encoding for a two–node network $A \rightarrow B$, where A has values $a, \bar{a}$ and B has values $b, \bar{b}$. The CNF encoding Δ has the following clauses:

$$\begin{aligned} \lambda_a \vee \lambda_{\bar{a}}, \quad & \neg\lambda_a \vee \neg\lambda_{\bar{a}} \\ \lambda_b \vee \lambda_{\bar{b}}, \quad & \neg\lambda_b \vee \neg\lambda_{\bar{b}} \\ \lambda_a &\leftrightarrow \theta_a \\ \lambda_{\bar{a}} &\leftrightarrow \theta_{\bar{a}} \\ \lambda_a \wedge \lambda_b &\leftrightarrow \theta_{b|a} \\ \lambda_a \wedge \lambda_{\bar{b}} &\leftrightarrow \theta_{\bar{b}|a} \\ \lambda_{\bar{a}} \wedge \lambda_b &\leftrightarrow \theta_{b|\bar{a}} \\ \lambda_{\bar{a}} \wedge \lambda_{\bar{b}} &\leftrightarrow \theta_{\bar{b}|\bar{a}}. \end{aligned} \tag{7}$$

This CNF has four models and encodes the multi-linear function:

$$f = \lambda_a\lambda_b\theta_a\theta_{b|a} + \lambda_a\lambda_{\bar{b}}\theta_a\theta_{\bar{b}|a} + \lambda_{\bar{a}}\lambda_b\theta_{\bar{a}}\theta_{b|\bar{a}} + \lambda_{\bar{a}}\lambda_{\bar{b}}\theta_{\bar{a}}\theta_{\bar{b}|\bar{a}}.$$

Note that a variable such as $\theta_{b|\bar{a}}$ is interpreted as a propositional variable when it appears in the CNF Δ, but is interpreted as a real variable with values in $[0, 1]$ when it appears in the multi–linear function f.

The encoding given in Definition 3 depends only on the network structure and the domains of its variables. Hence, two networks with the same structure and domains will generate the same encoding. We show next how to refine this encoding in order to exploit non–structural network properties: logical constraints and context–specific independence.

Logical constraints. A logical constraint corresponds to a conditional probability which is equal to either 0 or 1. One can produce a more efficient CNF encoding in the presence of logical constraints. Suppose for example that a network parameter $\theta_{x|u_1,\ldots,u_m}$ takes the value 0. We can then replace Clauses 3–5 by a single clause:

$$\neg\lambda_x \vee \neg\lambda_{u_1} \vee \ldots \vee \neg\lambda_{u_m}.$$

Hence, the variable $\theta_{x|u_1,\ldots,u_m}$ is eliminated from the encoding and the number of clauses is reduced. We can do this since every term which includes the indicators $\lambda_x, \lambda_{u_1}, \ldots, \lambda_{u_m}$ is multiplied by 0. Since these terms do not contribute to the multi–linear function value, there is no harm in excluding them. The above clause achieves this effect by excluding models that encode these terms. Similarly, if $\theta_{x|u_1,\ldots,u_m} = 1$, we can omit the parameter $\theta_{x|u_1,\ldots,u_m}$ and its associated clauses 3–5 from the encoding. The clauses only say that every term which includes the indicators $\lambda_x, \lambda_{u_1}, \ldots, \lambda_{u_m}$ is multiplied by $1 = \theta_{x|u_1,\ldots,u_m}$. It is then safe to eliminate this parameter and the corresponding clauses as they are vacuous in this case. As we show in Section 5, not only do logical constraints lead to a more efficient encoding, but they also lead to a major reduction in the size of compiled d-DNNF and its corresponding arithmetic circuit. This is a major advantage of our proposed approach for probabilistic inference as it can exploit logical constraints computationally, without requiring any algorithmic refinements. The exploitation takes place at the encoding level.

Context–specific independence. Suppose we have a variable X with two parents Y and Z in a belief network. Suppose further that $\theta_{x|yz} = \theta_{x|y\bar{z}}$. This means that given y, our belief in x is independent of Z. This form of independence is known as *context–specific independence (CSI)* [3] and is quite different from *structural independence* since we cannot identify it by examining the network structure. There is evidence that CSI is very valuable computationally [5, 31], yet it has proven hard to exploit it in the context of pure structure–based algorithms, such as the jointree algorithm [18, 17]. One can exploit CSI quite easily in the proposed framework. Specifically, all we have to do in the previous case is to replace the parameters $\theta_{x|yz}$ and $\theta_{x|y\bar{z}}$ in the CNF encoding by a new parameter, $\theta_{x|y}$. Moreover, we replace the clauses for these two parameters with $\lambda_x \wedge \lambda_y \leftrightarrow \theta_{x|y}$. We show in Section 5 the dramatic effect that CSI has on the size of d-DNNF and its corresponding arithmetic circuit.

We close this section by stressing that the CNF encoding of a belief network can be generated automatically once we know the network structure and the domain of each network variable. To incorporate logical constraints, however, we also need to know whether a network parameter attains the value 0 or 1. And to incorporate context–specific independence, we need to known which network parameters are equal.

4.3 A new approach to probabilistic inference

We now summarize our proposed approach for inference in belief networks. Given a belief network N, our

approach amounts to the following steps:

1. Encode the multi–linear function of N using a propositional theory Δ as given in Definition 3. Again, this can be completely automated with no need for human intervention.
2. Compile the encoding Δ into a smooth d-DNNF Γ using a compiler such as the one in [10]. One can use an OBDD package for this purpose too since OBDDs are a special case of d-DNNFs.[4] OBDDs are less succinct though, leading to arithmetic circuits which are larger than is really needed.[5]
3. Extract an arithmetic circuit Υ from the smooth d-DNNF Γ as given in Definition 2.
4. Use the circuit Υ for linear-time inference as described in [7, 6].

We present experimental results in the following section, which illustrate the effectiveness of our proposed approach.

We close this section by mentioning two other approaches for generating circuit implementations of the multi–linear functions corresponding to belief networks. First, we have shown in [7] that if we have a belief network with n variables, and an elimination order of width w for the network, then we can generate an arithmetic circuit that implements the network multi–linear function in $O(n \exp(w))$ time and space.[6] Second, we have shown recently that a carefully constructed jointree and a choice of one of its clusters can be viewed as an implicit circuit implementation of the network multi–linear function (where separator entries correspond to addition nodes and cluster entries correspond to multiplication nodes) [23]. Moreover, we have shown that a message-collect towards the chosen cluster corresponds to a circuit evaluation, while a message-distribute from the cluster corresponds to a circuit differentiation. Hence, the jointree method can be viewed as a special case of the framework proposed here, where the construction of a jointree corresponds to a specific method for constructing circuit implementations of the network multi–linear function. The arithmetic circuits generated by jointrees are interesting as they have a "regular" structure, allowing one to represent them without having to explicitly represent their edges (edges can be induced from the jointree structure). Using jointrees however (and any other structure–based method) for generating circuits is much less attractive in the presence of logical constraints and context–specific independence. More on this point in the following section.

[4] We also need to smooth the OBDD, which can be done in polynomial time [9].

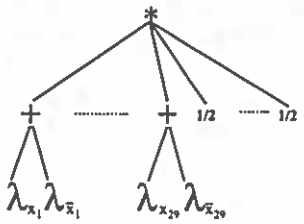
[5] The arithmetic circuit extracted from an OBDD is closely related to an algebraic decision diagram (ADD) [2]. See also [22, 16] for proposals on using OBDDs and ADDs as part of algorithms for probabilistic inference.

[6] One way to define the *width* of an elimination order is as the size of the maximum clique -1 in the jointree constructed based on the order.

5 Experimental results

We have two main goals in this section. Our first goal is to highlight the difference between our proposed approach for inference in belief networks, and structure–based approaches, such as those based on jointrees. In particular, we will present a few specific examples which illustrate the extent to which the proposed approach is sensitive to non–structural properties of belief networks: logical constraints and context–specific independence. Our second goal in this section is to provide statistics on the arithmetic circuits we were able to build for belief networks that are completely outside the scope of structure–based approaches. These networks correspond to digital circuits, with jointrees that have clique sizes > 60 (the complexity of the jointree algorithm is exponential in the size of maximum clique size).

We start with an example that we found to be quite revealing: the *printer_ts* belief network, distributed with the evaluation version of HUGIN (*www.hugin.com*). The network has 29 binary variables, call them $X_1, \ldots, X_{29}$, and 272 parameters. HUGIN compiles the network into a jointree with 11 cliques and 10 separators. The total size of clique tables is 236, and the total size of separator tables is 20. This means that if we were to construct an arithmetic circuit based on this jointree, as described in [23], the circuit will have 236 multiplication nodes and 20 addition nodes. The circuit we obtained using our approach had only 1 multiplication node and 29 addition nodes:



This appeared surprising at first, but it turns out that all parameters in the *printer_ts* network are actually set to 1/2, which means that the network is full of context–specific independence (this is probably not intended). Hence, even though the multi–linear function

of the network has 2^{29} terms, the function has a simple factorization: $(1/2)^{29}\prod_{i=1}^{29}(\lambda_{x_i}+\lambda_{\bar{x}_i})$. In fact, we would obtain this factorization, which size is linear in the number of network variables, regardless of the belief network structure. This example shows that our approach can generate a circuit of linear size for a belief network of arbitrary structure, if enough context-specific independence exists in the network.

Our second set of examples concern the extent to which logical constraints in a belief network affect the size of its compiled arithmetic circuit. Here we consider a few networks which are distributed with the evaluation version of HUGIN: *poker*, *golf*, *boblo*, *6nt*, *6hj* and *3nt*. Each one of these networks has a large number of 0/1 parameters. For example, the network *golf* has 8 multi-valued nodes, three of which have CPTs with only 0/1 parameters. We show below the size of arithmetic circuit we obtained for each network (number of addition/multiplication nodes), using the logic-based method proposed in this paper, in addition to the structure-based method based on jointrees [23]. The jointrees were computed by Netica (*www.norsys.com*).

Net	Vars#	d-DNNF-based *⁄+nodes	Jointree-based *⁄+nodes
Poker	7	302	685
Golf	8	143	676
Boblo	22	393	494
6nt	58	1377	12378
6hj	58	1814	29176
3nt	58	6328	35902

As is clear from the above table, logical constraints lead to significant reductions in the size of arithmetic circuits, and our proposed method appears to naturally exploit these constraints.

The computational value of logical constraints to probabilistic reasoning has long been observed. In fact, even structure-based methods have been refined to exploit such constraints. For example, a method known as *zero compression* has been proposed for jointrees [19], which reduces the size of a jointree in the presence of logical constraints.

We now consider a number of real-world belief networks which correspond to digital circuits that have very high connectivity. The goal here is to reason about the probability that any particular wire will have a certain signal on it, given evidence about other wires in the circuit. All we are given, in addition to the circuit, is a probability distribution over each circuit input. A belief network for this reasoning application is constructed in a systematic way, by including a network variable for each wire in the circuit and then adding an edge from each gate's input to its output. The CPT entries for each root node X in the network—that is, θ_x and $\theta_{\bar{x}}$—are determined based on the given distribution for the corresponding circuit input. All other CPT entries are either 0 or 1, and are determined based on the gate corresponding to that CPT. Belief networks which have 0/1 parameters everywhere, except possibly for the parameters of root variables, are known as *deterministic belief networks*. Such networks appear extensively in applications relating to causal and diagnostic reasoning [25]. A belief network where every node is related to its parents by a *noisy-or* model can also be easily reduced to a deterministic belief network [24].

We can build arithmetic circuits for deterministic belief networks as described in the previous section, but there is a much more efficient encoding method which we describe and use next.

Definition 4 *Let N be a deterministic belief network with variables $X_1,\ldots,X_n$, where each variable has two values: positive and negative. The CNF encoding of network N is a propositional theory over variables $X_1,\ldots,X_n$, which includes the clause $l(x)\vee l(u_1)\vee\ldots l(u_m)$ iff the parameter $\theta_{x|u_1,\ldots,u_m}$ equals 0. Here, $l(x)=\neg X$, when x is the positive value of variable X, and $l(x)=X$, when x is the negative value of X.*

This encoding is much more efficient than the one given in Definition 3 as it only includes one propositional variable for every network variable. It also includes a smaller number of clauses. Extracting an arithmetic circuit from this encoding is also a bit different from what is given in Definition 2.

Definition 5 *Let Δ be a smooth d-DNNF which encodes a deterministic belief network N over variables $X_1,\ldots,X_n$. Let x_i denote the positive value of X_i and $\bar{x}_i$ denote its negative value. The arithmetic circuit encoded by Δ is obtained by replacing every conjunction with a multiplication; every disjunction with an addition; every leaf node X_i with $\lambda_{x_i}*\theta_{x_i}$ if X_i is a root in N, and with λ_{x_i} otherwise; every leaf node $\neg X$ with $\lambda_{\bar{x}_i}*\theta_{\bar{x}_i}$ if X is a root in N, and with $\lambda_{\bar{x}_i}$ otherwise.*

Theorem 3 *The arithmetic circuit described in Definition 5 implements the multi-linear function of the corresponding deterministic belief network.*

Consider a circuit which consists of only one exclusive-or gate, leading to the structure $X_1\rightarrow X_3\leftarrow X_2$. The CNF encoding of this network will be as follows:

$$\neg X_1\vee\neg X_2\vee\neg X_3$$
$$X_1\vee X_2\vee\neg X_3$$

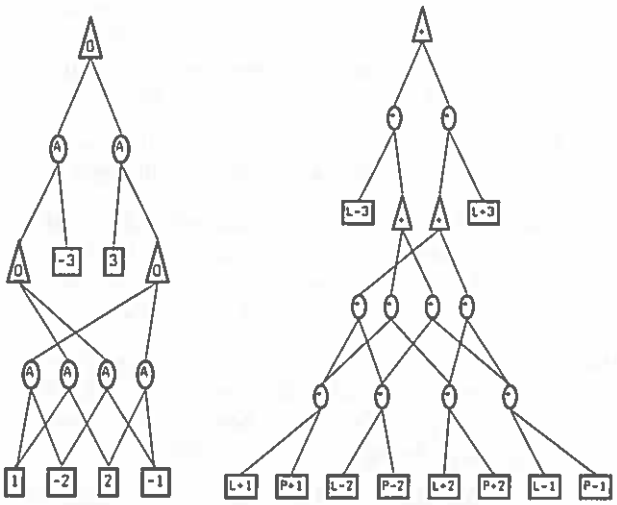


Figure 4: On the left: a smooth d-DNNF which encodes a deterministic belief network $X_1 \rightarrow X_3 \leftarrow X_2$, where X_3 is an exclusive-or of X_1 and X_2. On the right: an arithmetic circuit extracted from the d-DNNF. Here, L+i represents λ_{x_i}, L-i represents $\lambda_{\bar{x}_i}$, P+i represents θ_{x_i}, and P-i represents $\theta_{\bar{x}_i}$.

$$\neg X_1 \vee X_2 \vee X_3$$
$$X_1 \vee \neg X_2 \vee X_3$$

A smooth d-DNNF for this encoding is given in Figure 4(a), and the arithmetic circuit extracted from this d-DNNF is given in Figure 4(b). Let us now see how this circuit can be used to answer queries. Suppose the distribution over input X_1 is $\theta_{x_1} = .4, \theta_{\bar{x}_1} = .6$, and the distribution over input X_2 is $\theta_{x_2} = .7, \theta_{\bar{x}_2} = .3$. Let us now compute the probability $Pr(X_3 = \bar{x}_3)$. We can do this by evaluating the circuit under the following inputs $\lambda_{x_1} = \lambda_{\bar{x}_1} = \lambda_{x_2} = \lambda_{\bar{x}_2} = \lambda_{\bar{x}_3} = 1$ and $\lambda_{x_3} = 0$. The circuit evaluates to .46 under these inputs, which is the probability of $X_3 = \bar{x}_3$. One can answer a number of other queries based on the derivatives of the circuit, but we refer the reader to [7, 6] for details.

We now turn to some real-world deterministic belief networks, which correspond to combinational and sequential circuits selected from the ISCAS85 and ISCAS89 suites [1].[7] Table 1 reports on the sizes of d-DNNFs we were able to obtain for the CNF encodings of some members of these suites—more details

[7]Sequential circuits have been converted into combinational circuits in a standard way, by cutting feedback loops into flip-flops, treating a flip-flop's input as a circuit output and its output as a circuit input.

Network	Vars#	d-DNNF nodes#	d-DNNF edge#	Max Clique	d-DNNF Time (s)
c432	196	2899	19779	28	6
c499	243	691803	2919960	23	448
c880	443	3975728	7949684	24	1893
c1355	587	338959	3295293	23	809
c1908	913	6183489	12363322	45	5712
s510	236	967	5755	38	2
s953	440	11542	110266	64	14
s967	439	20645	443233	60	117
s1196	561	12554	261402	51	60
s1238	540	14512	288143	53	58
s1488	667	6338	62175	49	11
s1494	661	6827	64888	51	12
s1512	866	12560	140384	21	27
s3330	1961	358093	8889410	43	5853
s3384	1911	44487	392223	17	45

Table 1: Experimental results on deterministic belief networks.

can be found in [10]. An arithmetic circuit can be extracted from a given d-DNNF by first smoothing the d-DNNF, which increases its size slightly, and then applying the method of Definition 5 in a straightforward manner.[8] Some of the circuits in Table 1 are notorious for their high connectivity [14]. Table 1 lists the circuits we experimented with, in addition to the size of largest clique in the best jointree we could construct for their corresponding belief networks, using both classical methods [20] and more recent ones [11]. As is obvious from the reported clique sizes, these networks are quite hard for structure-based algorithms, such as the jointree algorithm. Moreover, some of them are completely outside the scope of these algorithms, since their complexity is exponential in clique sizes.

The success of our proposed approach on deterministic belief networks appears to be consistent with findings reported in [22], where OBDDs were used to encode a special class of belief networks that arise in troubleshooting applications. The approach in [22] was found to be quite favorable when compared with two implementations of the jointree algorithm: the Shenoy-Shafer [29] and the HUGIN architectures [28]. We note, however, that a number of the CNF encodings corresponding to networks in Table 1 could not be compiled into OBDDs, using the state-of-the-art CUDD package (*http://vlsi.colorado.edu/~fabio/CUDD/*), even though they were compiled successfully into d-DNNFs [10].

6 Conclusion

We have recently proposed a tractable logical form, known as deterministic, decomposable negation nor-

[8]To smooth a d-DNNF, we visit each or-node n and each of its children c. If $Vars(n) \neq Vars(c)$, we replace the child c of n by a new child: $c \wedge \bigwedge_{X \in Vars(n) - Vars(c)} (X \vee \neg X)$. Smoothing is an equivalence-preserving operation.

mal form (d-DNNF). We have shown that d-DNNF supports a number of logical operations in polynomial time, including clausal entailment, model counting, model enumeration, model minimization, and probabilistic equivalence testing. In this paper, we discussed another major application of this logical form: the implementation of multi-linear functions (of exponential size) using arithmetic circuits (that are not necessarily exponential). Specifically, we showed that each multi-linear function can be encoded using a propositional theory, and that each d-DNNF of the theory corresponds to an arithmetic circuit that implements the encoded multi-linear function. We discussed the application of these results to factoring belief networks, which can be viewed as multi-linear functions as has been shown recently. We also discussed the merits of the proposed approach for factoring belief networks, and presented experimental results showing how it can handle efficiently belief networks that are intractable to structure-based methods for probabilistic inference.

Acknowledgement

This work has been partially supported by NSF grant IIS-9988543 and MURI grant N00014-00-1-0617.

References

[1] ISCAS Benchmark Circuits, http://www.cbl.ncsu.edu/www/CBL_Docs.

[2] R. Bahar, E. Frohm, C. Gaona, G. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic decision diagrams and their applications. In *ACM/IEEE International Conference on Computer Aided Design*, 1993.

[3] Craig Boutilier, Nir Friedman, Moisés Goldszmidt, and Daphne Koller. Context-specific independence in bayesian networks. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 115–123, 1996.

[4] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35:677–691, 1986.

[5] B.D. D'Ambrosio. Local Expression Languages for Probabilistic Dependence. *International Journal of Approximate Reasoning*, 13(1):61–81, 1995.

[6] Adnan Darwiche. A differential approach to inference in Bayesian networks. Technical Report D–108, Computer Science Department, UCLA, Los Angeles, Ca 90095, 1999. To appear in the Journal of ACM.

[7] Adnan Darwiche. A differential approach to inference in Bayesian networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 123–132, 2000.

[8] Adnan Darwiche. Decomposable negation normal form. *Journal of the ACM*, 48(4):1–40, 2001.

[9] Adnan Darwiche. On the tractability of counting theory models and its application to belief revision and truth maintenance. *Journal of Applied Non-Classical Logics*, 11(1-2):11–34, 2001.

[10] Adnan Darwiche. A compiler for deterministic decomposable negation normal form. Technical Report D–125, Computer Science Department, UCLA, Los Angeles, Ca 90095, 2002.

[11] Adnan Darwiche and Mark Hopkins. Using recursive decomposition to construct elimination orders, jointrees and dtrees. In *Trends in Artificial Intelligence, Lecture notes in AI, 2143*, pages 180–191. Springer-Verlag, 2001.

[12] Adnan Darwiche and Jinbo Huang. Testing equivalence probabilistically. Technical Report D–123, Computer Science Department, UCLA, Los Angeles, Ca 90095, 2002.

[13] Adnan Darwiche and Pierre Marquis. A perspective on knowledge compilation. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 175–182, 2001.

[14] Yousri El Fattah and Rina Dechter. An evaluation of structural paramters for probabilistic reasoning: Results on benchmark circuits. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 244–251, 1996.

[15] J. Gergov and C. Meinel. Efficient analysis and manipulation of OBDDs can be extended to FBDDs. *IEEE Transactions on Computers*, 43(10):1197–1209, 1994.

[16] Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutilier. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 279–288, 1999.

[17] Cecil Huang and Adnan Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, 1996.

[18] F. V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in recursive graphical models by local computation. *Computational Statistics Quarterly*, 4:269–282, 1990.

[19] Frank Jensen and Stig K. Andersen. Approximations in Bayesian belief universes for knowledge based systems. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 162–169, Cambridge, MA, July 1990.

[20] U. Kjaerulff. Triangulation of graphs—algorithms giving small total state space. Technical Report R-90-09, Department of Mathematics and Computer Science, University of Aalborg, Denmark, 1990.

[21] Z. Li and B.D. D'Ambrosio. Efficient Inference in Bayes Networks as a Combinatorial Optimization Problem. *International Journal of Approximate Reasoning*, 11(1):55–81, 1994.

[22] T. Nielsen, P. Wuillemin, F. Jensen, and U. Kjaerulff. Using ROBDDs for inference in Bayesian networks with troubleshooting as an example. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 426–435, 2000.

[23] James Park and Adnan Darwiche. A differential semantics for jointree algorithms. Technical Report D–118, Computer Science Department, UCLA, Los Angeles, Ca 90095, 2001.

[24] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988.

[25] Judea Pearl. *Causality.* Cambridge University Press, 2000.

[26] David Poole. Context-specific approximation in probabilistic inference. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 447–454, 1998.

[27] Graz Rote. Path problems in graphs. *Computing Suppl.*, 7:155–189, 1990.

[28] F. Jensen S. Anderson, K. Olesen and F. Jensen. Hugin – a shell for building beliefs universes for expert systems. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 1989.

[29] Prakash P. Shenoy and Glenn Shafer. Propagating belief functions with local computations. *IEEE Expert*, 1(3):43–52, 1986.

[30] J. von zur Gathen. Algebraic complexity theory. *Ann. Rev. Comp. Sci.*, 3:317–347, 1988.

[31] Nevin Lianwen Zhang and David Poole. Exploiting causal independence in bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328, 1996.

A Proofs

Proof of Theorem 1

We first define the function *Decode1* for a propositional theory Δ over variables Σ:

$$Decode1(\Delta, \Sigma) \stackrel{def}{=} \sum_{\sigma \models \Delta} \prod_{V \in \Sigma} \begin{cases} 1, & \text{if } \sigma(V) = false; \\ V, & \text{if } \sigma(V) = true. \end{cases}$$

Clearly, $Decode1(\Delta, \Sigma)$ returns the multi–linear function encoded by Δ according to Definition 1.

We now have the following properties of *Decode1*.

Determinism and Smoothness. When $\alpha \wedge \beta$ is inconsistent and $Vars(\alpha) = Vars(\beta) = \Sigma$:

$$Decode1(\alpha \vee \beta, \Sigma) = Decode1(\alpha, \Sigma) + Decode1(\beta, \Sigma).$$

This follows immediately from the definition of *Decode1*.

Decomposability. When $Vars(\alpha) \cap Vars(\beta) = \emptyset$ and $Vars(\alpha) \cup Vars(\beta) = \Sigma$:

$$\begin{aligned} &Decode1(\alpha \wedge \beta, \Sigma) \\ &\quad = Decode1(\alpha, Vars(\alpha)) \star Decode1(\beta, Vars(\beta)). \end{aligned}$$

If f_α is the multi–linear function encoded by α, and if f_β is the multi-linear function encoded by β, then $f_\alpha \star f_\beta$ is the multi–linear function encoded by $\alpha \wedge \beta$ in this case.

Negative literal. $Decode1(\neg V, \{V\}) = 1$. This follows because $\neg V$ has only one model which sets its only variable V to *false*. Hence, $\neg V$ encodes a multi–linear function f that contains one term that has no variables; $f = 1$.

Positive literal. $Decode1(V, \{V\}) = V$. This follows because V has only one model which sets its only variable V to *true*. Hence, V encodes a multi–linear function that contains one term V; $f = V$.

The arithmetic circuit given in Definition 2 is then a trace of the application of function *Decode1* to $\Delta(n)$, where n is the root of smooth d-DNNF Δ. □

Proof of Theorem 2

We prove that the described CNF Δ does indeed encode the multi–linear function f of the given belief

network by showing a one-to-one correspondence between the models of Δ and the terms of f.

Suppose that t is a term in the multi-linear function f. Let σ be the model that encodes t as given by Definition 1. We want to show that $\sigma \models \Delta$. The term t contains exactly one indicator λ_x for each variable X, where x is a value of X. Moreover, it contains exactly one parameter $\theta_{x|u_1,\ldots,u_m}$ for variable X, where the indicators $\lambda_x, \lambda_{u_1}, \ldots, \lambda_{u_m}$ appear in term t. The model σ will then set the indicator λ_x to *true* and all other indicators of X to *false*. It will also set the parameter $\theta_{x|u_1,\ldots,u_m}$ to *true* and all other parameters of X to *false*. The model σ will then clearly satisfy clauses 1-5 and, hence, $\sigma \models \Delta$.

Suppose now that we have a model $\sigma \models \Sigma$, and let t be the term encoded by σ as given by Definition 1. We want to show that t is a term in the multi-linear function f. Since the model σ satisfies clauses 1-2, it must then set exactly one indicator λ_x to true for every variable X. The term t will then include exactly one indicator for each variable X. Moreover, since the model σ satisfies clauses 3-5, it must set to true exactly one parameter for X: $\theta_{x|u_1,\ldots,u_m}$, where $\lambda_x, \lambda_{u_1}, \ldots, \lambda_{u_m}$ are the indicators of $X, U_1, \ldots, U_m$ that are set to true by σ. Hence, $\theta_{x|u_1,\ldots,u_m}$ will be the only parameter of X which is included in the term t. Hence, t is a term in the multi-linear function f. □

Proof of Theorem 3

First, note that the multi-linear function of given belief network has 2^n terms corresponding to the instantiations of network variables. A number of these terms, however, are multiplied by zero parameters and can be excluded. Each term which is not multiplied by a zero conditional parameter, has the form: $\lambda_{x_1}\theta_{x_1}\ldots\lambda_{x_m}\theta_{x_m}\lambda_{x_{m+1}}\ldots\lambda_{x_n}$, where $X_1, \ldots, X_m$ are the root variables in the belief network. This follows since all conditional parameters that are consistent with such a term must be equal to 1.

Let us now define the function *Decode2* for a propositional theory Δ over variables Σ and variables $\Gamma \subseteq \Sigma$:

$$\begin{aligned} &Decode2(\Delta, \Sigma, \Gamma) \\ &\stackrel{def}{=} \sum_{\sigma \models \Delta} \prod_{X \in \Sigma - \Gamma} \begin{cases} \lambda_x, & \text{if } \sigma(X) = true \\ \lambda_{\bar{x}}, & \text{if } \sigma(X) = false \end{cases} \\ &\qquad \prod_{X \in \Gamma} \begin{cases} \lambda_x \theta_x, & \text{if } \sigma(X) = true \\ \lambda_{\bar{x}} \theta_{\bar{x}}, & \text{if } \sigma(X) = false. \end{cases} \end{aligned}$$

If Δ is the CNF encoding a deterministic belief network N, and if N has variables Σ and root variables Γ, then $Decode2(\Delta, \Sigma, \Gamma)$ is clearly the multi-linear function of network N. This follows because the models of Δ are in one-to-one correspondence with the non-vanishing terms of the multi-linear function of N.

We now have the following properties of *Decode2*.

Determinism and Smoothness. When $\alpha \wedge \beta$ is inconsistent and $Vars(\alpha) = Vars(\beta) = \Sigma$:

$$\begin{aligned} &Decode2(\alpha \vee \beta, \Sigma, \Gamma) \\ &= Decode2(\alpha, \Sigma, \Gamma) + Decode2(\beta, \Sigma, \Gamma). \end{aligned}$$

Decomposability. When $Vars(\alpha) \cap Vars(\beta) = \emptyset$ and $Vars(\alpha) \cup Vars(\beta) = \Sigma$:

$$\begin{aligned} &Decode2(\alpha \wedge \beta, \Sigma, \Gamma) \\ &= Decode2(\alpha, Vars(\alpha), \Gamma \cap Vars(\alpha)) * \\ &\quad Decode2(\beta, Vars(\beta), \Gamma \cap Vars(\beta)). \end{aligned}$$

Negative literal.

$$Decode2(\neg V, \{V\}, \Gamma) = \begin{cases} \lambda_{\bar{x}} \theta_{\bar{x}}, & \text{if } X \in \Gamma; \\ \lambda_{\bar{x}}, & \text{otherwise.} \end{cases}$$

Positive literal.

$$Decode2(V, \{V\}, \Gamma) = \begin{cases} \lambda_x \theta_x, & \text{if } X \in \Gamma; \\ \lambda_x, & \text{otherwise.} \end{cases}$$

The arithmetic circuit given in Definition 5 is then a trace of the application of function *Decode2* to $\Delta(n)$, where n is the root of smooth d-DNNF Δ. □

Bipolar representation and fusion of preferences in the possibilistic logic framework

Salem Benferhat, Didier Dubois, Souhila Kaci and Henri Prade
Institut de Recherche en Informatique de Toulouse (I.R.I.T.)-C.N.R.S.
Université Paul Sabatier, 118 route de Narbonne 31062 TOULOUSE Cedex 4, FRANCE
E-mail:{benferhat, dubois, kaci, prade}@irit.fr

Abstract

Preferences are naturally bipolar: positive aspirations and rejections are complementary in order to express agents wishes. Possibility theory framework allows a simple bipolar representation, using the "classical" possibility and necessity functions, and also the less usual "guaranteed possibility" function. This paper first shows how to deal syntactically with positive aspirations as well as rejections. We then discuss the problem of merging multiple agents preferences in this bipolar framework both at semantic and syntactic levels. Lastly, we discuss the problem of the consistency between positive aspirations and rejections, and how to restore it when necessary.

1 Introduction

The problem of the representation of the preferences of agents has been considered by various researchers in Artificial Intelligence in the recent past years [23, 10, 5, 1, 18, 3, 6]. Indeed this issue is important when we have to represent the desires of users in information systems (e.g. recommender systems), or to reason about them and to solve conflicts between inconsistent goals, as e.g., in multi-agent systems.
Preferences are often expressed in two forms: positive aspirations and rejections. Indeed, on the one hand, an agent expresses what he considers as (more or less) impossible because it is unacceptable for him, and on the other hand he expresses what he considers as really satisfactory. For example assume that we have a one month summer school, and we ask some invited speaker to express his preference for scheduling his talk. We assume that the talks can be either given in the morning, or in the afternoon. The invited speaker may provide two kinds of preferences. In the first one, he specifies satisfactory slots, with satisfaction levels. This is positive preference. In the second one, he describes unacceptable slots with levels of tolerance. For instance, he may strongly refuse to participate one day (e.g., because it is the birthday of her daughter), and weakly refuse to speak on the last day. This is negative preferences or rejections.
This bipolar representation is supported by recent studies in cognitive psychology which have shown that positive and negative preferences are processed separately in the mind, and are felt as independent and different dimensions by people [8, 7]. Note that in general there is no symmetry between positive aspirations and rejections in the sense that positive aspirations do not just mirror what is not rejected.

The idea of bipolar representations of preferences has been considered in different frameworks, such as multicriteria decision making [16], or in Artificial Intelligence works oriented towards qualitative decision [22, 17]. However these works usually assume that the positive and negative parts of the preferences, once they have been specified and represented can be combined and processed together, rather than separately as it is done in this paper.

Section 2 gives the necessary background on possibility theory and possibilistic logic. Section 3 specifies the bipolar representation of preferences. Sections 4 and 5 discuss the possibilistic encoding of rejections and positive aspirations. Sections 6 and 7 present the problems of fusing of agents preferences and restoring the coherence when they conflict.

2 Background on possibility theory and possibilistic logic

We consider a propositional language $\mathcal{L}$ over a finite alphabet $\mathcal{P}$ of atoms. Ω denotes the set of all classical interpretations. $[\psi]$ denotes the set of models of the

proposition ψ.

The basic concept in possibilistic theory is the notion of possibility distribution, denoted by π, which is simply a function from the set of interpretations to the unit interval $[0,1]$, or to any totally ordered scale, finite or not. Thus we can use only a finite set of qualitative levels if necessary.

Given a posibility distribution π, two standard measures are defined for formulas:

- the possibility (or consistency) measure of a formula ϕ:

  $$\Pi(\phi) = max\{\pi(\omega) : \omega \models \phi\},$$

  which evaluates the extent to which ϕ is consistent with the available information expressed by π, and
- the necessity (or certainty) measure of a formula ϕ:

  $$N(\phi) = 1 - \Pi(\neg\phi),$$

  which evaluates the extent to which ϕ is entailed by information expressed by π.

Given a possibility distribution π, we define the core of π as the set of interpretations having the highest possibility degree in π. Formally,

Definition 1 *The core of a possibility distribution π, denoted by $core(\pi)$, is defined by:*

$$core(\pi) = \{\omega : \omega \in \Omega, \nexists \omega' \in \Omega, \pi(\omega') > \pi(\omega)\}.$$

We now define the contextual core as follows:

Definition 2 *The contextual core of a possibility distribution π given a formula φ (φ represents the context), denoted by $core_\varphi(\pi)$, is defined by:*

$$core_\varphi(\pi) = \{\omega : \omega \models \varphi, \nexists \omega', \omega' \models \varphi \text{ and } \pi(\omega') > \pi(\omega)\}.$$

Indeed, two kinds of inference can be defined from π, in the same spirit of [20]:

Definition 3 *Plausible and preferential inferences. Let π be a possibility distribution. The formula ψ is said to be a* **plausible** *consequence of π, denoted by $\pi \models_P \psi$, iff*

$$core(\pi) \subseteq [\psi].$$

ψ is said to be a **preferential** *consequence of π given the formula φ, denoted by $\pi \models_\varphi \psi$, iff*

$$core_\varphi(\pi) \subseteq [\psi].$$

Prioritized information are represented in the possibilistic logic framework by means of a set of weighted formulas, called *a possibilistic logic base*, of the form $\Sigma = \{(\phi_i, a_i) : i = 1, \cdots, n\}$, where ϕ_i is a propositional formula and a_i belongs to a totally ordered scale such as $[0,1]$. (ϕ_i, a_i) means that the necessity degree of ϕ_i is at least equal to a_i, i.e. $N(\phi_i) \geq a_i$.
Given a possibilistic logic base Σ, we can generate a unique possibility distribution, denoted by π_Σ, defined by [12]:

Definition 4 $\forall \omega \in \Omega$,

$$\pi_\Sigma(\omega) = \begin{cases} 1 & \text{if } \forall (\phi_i, a_i) \in \Sigma, \omega \models \phi_i \\ 1 - max\{a_i : (\phi_i, a_i) \in \Sigma \text{ and } \omega \not\models \phi_i\} & \text{otherwise,} \end{cases}$$

where π_Σ is the largest possibility distribution such that the necessity measure N associated with π_Σ is such that $N(\phi_i) \geq a_i$ holds for all i.
Besides, $N(\phi \vee \psi) \geq a, N(\neg\phi \vee \xi) \geq b \vdash N(\psi \vee \xi) \geq min(a,b)$, which is the basis of the possibilistic inference machinery [12].

3 Specifying bipolar preferences

In this paper, we propose a bipolar representation of preferences which can be expressed both at the syntactic and at the semantic levels. We introduce the syntactic specification of these preferences in this section. Preferences of an agent will be represented by two different sets of equality constraints.
The first set corresponds to what is not unacceptable, what is more or less acceptable, tolerable for the agent. It is of the form $\{\mathcal{R}(\phi_i) = a_i : i = 1, \cdots, n\}$ where ϕ_i is a propositional formula, $\mathcal{R}$ stands for rejection, and a_i is a real in the interval $[0,1]$. $\mathcal{R}(\phi_i) = a_i$ expresses the rejection strength of ϕ_i by the agent. $\mathcal{R}(\phi_i) = 1$ means that the agent strongly rejects ϕ_i, and any solution where ϕ_i is true is considered as not feasible by the agent. $\mathcal{R}(\phi_i) = 0$ simply means that ϕ_i is not rejected, and hence solutions satisfying ϕ_i are not at all unacceptable, or are feasible. For $\mathcal{R}(\phi_i) = a_i$ with $a_i \in (0,1)$, the higher a_i is, the less acceptable are the solutions satisfying ϕ_i. It turns out that the set of rejections can be easily handled using standard possibilistic logic based on the two classical functions of possibility theory: possibility and necessity measures. This will be developed in Section 4.
The second set expresses "positive" goals, or agent's aspirations. It is of the form $\{\mathcal{G}(\psi_j) = b_j : j = 1, \cdots, m\}$, where $\mathcal{G}$ stands for positive goals, ψ_j is a propositional formula, and b_j is a real in the interval $[0,1]$. $\mathcal{G}(\psi_j) = b_j$ expresses the minimal level of satisfaction which is guaranteed if ψ_j is true. $\mathcal{G}(\psi_j) = 1$ means that the agent is fully satisfied as soon as ψ_j

is true. $\mathcal{G}(\psi_j) = 0$ means that learning that ψ_j is satisfied brings no satisfaction to the agent who is indifferent. When $\mathcal{G}(\psi_j) = b_j$ for $b_j \in (0,1)$ the larger b_j is, the more satisfied is the agent if ψ_j is true. This kind of positive goals cannot be directly handled by the possibilistic logic machinery. In fact, positive goals can be represented using the so-called function of "guaranteed possibility", denoted by Δ, in possibility theory. We will show that the syntactic representation of uncertain information using Δ is dual in some sense of the one used in standard possibilistic logic. This will be developed in Section 5.

4 Modelling rejections in possibilistic logic

This section shows that possibilistic logic provides a natural framework for modelling rejections. Rejections can be represented, at the semantical level, by a total pre-order on the set of all possible outcomes, from what is feasible to what is considered as unacceptable by an agent. Outcomes are called here interpretations, since we use a logic-based representation. The total pre-order can be encoded, in possibility theory framework, using the notion of a possibility distribution, denoted by $\pi_{\mathbf{R}}$ ($\mathbb{R}$ for rejections), which is a function from the set of interpretations to the unit interval $[0,1]$. $\pi_{\mathbf{R}}(\omega)$ represents the degree of acceptability of ω given agent's preferences. $\pi_{\mathbf{R}}(\omega) = 1$ means that ω is fully acceptable, $\pi_{\mathbf{R}}(\omega) = 0$ means that ω is completely unacceptable (rejected), and more generally, $\pi_{\mathbf{R}}(\omega) > \pi_{\mathbf{R}}(\omega')$ means that ω' is more unacceptable than ω.

In practice an agent cannot provide the whole possibility distribution $\pi_{\mathbf{R}}$, but only a set of specifications of negative desires, and their level of rejection. Let $\mathbb{R} = \{\mathcal{R}(\phi_i) = a_i : i = 1, \cdots, n\}$ be this set of rejections, where by default, formulas which are not in $\mathbb{R}$ are assumed to have a rejection level equal to 0, namely they are not explicitely rejected at all.

The question is then how to compute $\pi_{\mathbf{R}}$. Let us illustrate the construction of $\pi_{\mathbf{R}}$ when we only have one strong rejection constraint $\mathcal{R}(\phi) = 1$. Let ω be an interpretation. Intuitively, if ω falsifies ϕ then ω is fully acceptable by the agent, i.e. $\pi_{\mathbf{R}}(\omega) = 1$, and if ω satisfies ϕ then ω is fully unacceptable, i.e. $\pi_{\mathbf{R}}(\omega) = 0$. Now, assume that ϕ is weakly rejected, i.e. $\mathcal{R}(\phi) = a < 1$. Again, if ω falsifies ϕ then it is fully acceptable. If ω satisfies ϕ then the higher is a, the less acceptable is ω. One way to achieve this constraint is to assign the value $1 - a$ to $\pi_{\mathbf{R}}(\omega)$. More formally, the possibility distribution associated with $\mathbb{R} = \{\mathcal{R}(\phi) = a\}$ is:

$$\pi_{\mathbf{R}}(\omega) = \begin{cases} 1 & \text{if } \omega \not\models \phi \\ 1 - a & \text{if } \omega \models \phi. \end{cases}$$

Now assume that the agent both rejects ϕ and ψ with $\mathcal{R}(\phi) = a$ and $\mathcal{R}(\psi) = b$ respectively. Then, we have three cases:

- $\omega \not\models \phi$ and $\omega \not\models \psi$, then ω is fully acceptable. Hence $\pi_{\mathbf{R}}(\omega) = 1$.
- $\omega \models \neg\phi \wedge \psi$ (resp. $\omega \models \phi \wedge \neg\psi$) then the higher is b (resp. a), the less acceptable is ω. Hence $\pi_{\mathbf{R}}(\omega) = 1 - b$ (resp. $1 - a$).
- $\omega \models \phi \wedge \psi$ then the higher is a or b, the less acceptable is ω. Hence $\pi_{\mathbf{R}}(\omega) = 1 - max(a, b)$.

More generally, we have:

Definition 5 *The possibility distribution $\pi_{\mathbf{R}}$ associated with a set of rejections $\mathbb{R} = \{\mathcal{R}(\phi_i) = a_i : i = 1, \cdots, n\}$ is:*

$\pi_{\mathbf{R}}(\omega) = 1 - max\{a_i : \omega \models \phi_i, \mathcal{R}(\phi_i) = a_i \in \mathbb{R}\}$, *with* $max\{\emptyset\} = 0$.

Clearly, this definition is very close to the one used in standard possibilistic logic, for inducing a possibility distribution π_Σ from a possibilistic knowledge base Σ, viewed as expressing constraints in terms of necessity measure. Indeed, it is enough to replace ϕ_i by $\neg\phi_i$ in $\mathbb{R}$ in order to obtain Σ. Hence, a set of rejections can be directly encoded by means of a possibilistic knowledge base:

Proposition 1 *Let $\mathbb{R} = \{\mathcal{R}(\phi_i) = a_i : i = 1, \cdots, n\}$. Let $\Sigma = \{(\neg\phi_i, a_i) : \mathcal{R}(\phi_i) = a_i \in \mathbb{R}\}$. Then, $\pi_{\mathbf{R}} = \pi_\Sigma$, where π_Σ is the semantic counterpart of the set Σ of possibilistic logic formulas.*

Proofs of Propositions are provided in Appendix.

This result is important since it means that the classical possibilistic logic machinery can be used for handling negative desires and drawing inferences from them. Moreover, the complexity of possibilistic logic is only slightly higher than the one of classical logic [12].

Example 1 *Let co, be and su be three propositional symbols which stand respectively for "country", "beach" and "sun". Assume that the agent provides the following set of rejections:*

$\mathbb{R} = \{\mathcal{R}(be \wedge \neg su) = 1, \mathcal{R}(co \wedge su) = 1,$
$\mathcal{R}(\neg be \wedge su) = \frac{1}{2}\}$.

ω	$\pi_{\mathbf{R}}(\omega)$
$\neg co \wedge \neg be \wedge \neg su$	1
$\neg co \wedge be \wedge su$	1
$co \wedge \neg be \wedge \neg su$	1
$\neg co \wedge \neg be \wedge su$	$\frac{1}{2}$
other interpretations	0

Table 1: The possibility distribution associated with $\mathbb{R}$.

This describes the rejections of an agent regarding sunny places: a total refusal of beach without sun, a total refusal of a country place where there is sun. The last rejection expresses a partial refusal to go to a sunny place where there is no beach.
Table 1 gives the possibility distribution associated with $\mathbb{R}$ applying Definition 5.

Note that the possibilistic base associated with $\mathbb{R}$ is the following $\Sigma = \{(\neg be \vee su, 1), (\neg co \vee \neg su, 1), (be \vee \neg su, \frac{1}{2})\}$. We can check that Σ generates the same possibility distribution as given in Table 1, using Appendix.

5 Representing positive goals

5.1 Semantic representation

The positive goals of an agent can also be described, at the semantical level, in terms of a possibility distribution also. Let $\pi_{\mathbf{G}}$ ($\mathbb{G}$ for goals) be this distribution. The range of $\pi_{\mathbf{G}}$ is also $[0, 1]$ (or any totally ordered set finite or not, and may be different from the one used for $\pi_{\mathbf{R}}$). $\pi_{\mathbf{G}}(\omega) > \pi_{\mathbf{G}}(\omega')$ means that ω is more satisfactory than ω'.
The meaning of $\pi_{\mathbf{G}}(\omega)$ is different from $\pi_{\mathbf{R}}(\omega)$. Indeed $\pi_{\mathbf{G}}(\omega)$ evaluates to what degree ω is satisfactory for the agent, while $\pi_{\mathbf{R}}(\omega)$ evaluates to what degree ω is acceptable, feasible by the agent. In particular $\pi_{\mathbf{G}}(\omega) = 1$ means that the agent is fully satisfied, while $\pi_{\mathbf{G}}(\omega) = 0$ simply means that the agent is indifferent (while $\pi_{\mathbf{R}}(\omega) = 0$ means that ω is impossible).

As for $\mathbb{R}$, in practice, an agent provides a set of positive goals of the form $\mathbb{G} = \{\mathcal{G}(\psi_j) = b_j : j = 1, \cdots, m\}$, where by default it is assumed that the agent is indifferent with respect to the truth of propositions which are not explicitely stated in $\mathbb{G}$. $\mathcal{G}(\psi_j) = b_j$ means that the agent will be satisfied to a degree b_j if ψ_j alone is satisfied.

In a similar way, let us see how to build the possibility distribution, denoted by $\pi_{\mathbf{G}}$, associated with $\mathbb{G}$. We first consider the case where we only have one constraint $\mathcal{G}(\psi) = b$. Then if a given ω satisfies ψ, the associated level of satisfaction will be equal to b. Otherwise, this level will be equal to 0, namely the agent is indifferent (and not to $1 - b$ which corresponds to the rejection of $\neg\psi$). In the general case, given a set of positive goals, the level of satisfaction associated with ω is equal to the highest level of a formula appearing in $\mathbb{G}$ satisfied by ω:

ω	$\pi_{\mathbf{G}}(\omega)$
$\neg co \wedge be \wedge su$	1
$co \wedge \neg be \wedge \neg su$	$\frac{1}{4}$
other interpretations	0

Table 2: The possibility distribution associated with $\mathbb{G}$.

Definition 6 *The possibility degree $\pi_{\mathbf{G}}$ associated with a set of positive goals $\mathbb{G} = \{\mathcal{G}(\psi_j) = b_j : j = 1, \cdots, m\}$ is:*

$\pi_{\mathbf{G}}(\omega) = max\{b_j : \omega \models \psi_j \text{ and } \mathcal{G}(\psi_j) = b_j \in \mathbb{G}\}$, *with* $max\{\emptyset\} = 0$.

Note that the addition of positive goals in $\mathbb{G}$ can only lead to the increasing of the satisfaction level associated with ω. Let us emphasize that this contrasts with the behaviour of $\pi_{\mathbf{R}}$ which is monotonically decreasing with respect to the number of constraints in $\mathbb{R}$.

Example 2 *Let $\mathbb{G}$ be the following set of goals $\mathbb{G} = \{\mathcal{G}(\neg co \wedge be \wedge su) = 1, \mathcal{G}(co \wedge \neg be \wedge \neg su) = \frac{1}{4}\}$. The first expression means that the agent is fully satisfied when there is a sun, a beach and he is not in the country. The second expression means that the agent is weakly satisfied when he is in the country (without beach or sun).*
The possibility distribution $\pi_{\mathbf{G}}$ associated with $\mathbb{G}$ is given in Table 2 following Definition 6.

5.2 Properties of the representation

The set of positive goals, contrarily to the rejections, cannot be directly handled by standard possibilistic logic. This is due to the fact that $\mathcal{G}(\psi) = b$ cannot be directly expressed using possibility and necessity measures as $\mathbb{R}$. Rather constraints like $\mathcal{G}(\psi) = b$ have to be represented using a third function called *guaranteed possibility* function, denoted by Δ [15]. The expression $\Delta(\psi) = b$ means that any interpretation where ψ is true, has a satisfactory degree at least equal to b. This is exactly what is intended by the information encoded in $\mathbb{G}$, indeed $\Delta(\psi) = min_{\omega \models \psi} \pi_{\mathbf{G}}(\omega)$[1].

[1] Hence $\Delta(\phi \vee \psi) = min(\Delta(\phi), \Delta(\psi))$, so Δ is decreasing with respect to entailment. Indeed, the semantic en-

Moreover, there exists few works where the logical machinery of the Δ-weighted formulas is described. It is governed by a cut rule of the form $\Delta(\phi \wedge \psi) \geq a, \Delta(\neg\phi \wedge \xi) \geq b \vdash \Delta(\psi \wedge \xi) \geq min(a,b)$ [11] which is the counterpart of the possibilistic resolution changing Δ into a necessity measure and the conjunctions into disjunctions.

Let us now give some properties of this representation:

Lemma 1 *Let $\mathbb{G}$ be a set of positive goals containing ψ_1 and ψ_2 at the level b. Let $\mathbb{G}'$ be a set of positive goals obtained from $\mathbb{G}$ by replacing $\mathcal{G}(\psi_1) = b$ and $\mathcal{G}(\psi_2) = b$ by $\mathcal{G}(\psi_1 \vee \psi_2) = b$. Then, $\mathbb{G}$ and $\mathbb{G}'$ are semantically equivalent, i.e. $\pi_{\mathbf{G}} = \pi_{\mathbf{G}'}$.*

This means that two sets of goals having the same strength can be replaced by their disjunction with a same strength. The second lemma concerns redundant information:

Definition 7 *$\mathcal{G}(\phi) = a \in \mathbb{G}$ is said to be subsumed if there exists $\mathcal{G}(\psi) = b \in \mathbb{G}$ such that $b \geq a$ and $\phi \vdash \psi$.*

Lemma 2 *Let $\mathcal{G}(\phi) = a$ be a subsumed goal in $\mathbb{G}$. Let $\mathbb{G}' = \mathbb{G} - \{\mathcal{G}(\phi) = a\}$. Then, $\mathbb{G}$ and $\mathbb{G}'$ are semantically equivalent, i.e. $\pi_{\mathbf{G}} = \pi_{\mathbf{G}'}$.*

In the following, we define the consistency degree of a set of goals, which is the dual of the inconsistency degree of a possibilistic knowledge base:

Definition 8 *Let $\mathbb{G}$ be a set of positive goals. The consistency degree of $\mathbb{G}$, denoted by $Cons(\mathbb{G})$, is defined by:*

$$Cons(\mathbb{G}) = max\{a_i : \mathcal{G}(\phi_i) = a_i \in \mathbb{G} \text{ and } \phi_i \text{ is consistent}\}.$$

Indeed, we define the consistent subbase of $\mathbb{G}$ for making inferences as follows:

Definition 9 *Let $\mathbb{G}$ be a set of positive goals. The preferred consistent subbase of $\mathbb{G}$, denoted by $\rho_\Delta(\mathbb{G})$, is:*

$$\rho_\Delta(\mathbb{G}) = \{\phi_i : \mathcal{G}(\phi_i) = a_i \in \mathbb{G} \text{ and } a_i = Cons(\mathbb{G})\}.$$

Given these two definitions, we are now able to provide the syntactic counterpart of the plausible and preferential inference:

tailment goes here in a way which is the opposite of the situation in the classical logical representation framework. Namely, if all the models of ϕ are feasible, we can conclude from this piece of information that all the worlds in ψ are feasible only if the entailment $\psi \models \phi$ holds, and nothing is said about the interpretations outside the models of ϕ.

Proposition 2

1. *Let $\mathbb{G}$ be a set of positive goals and $\pi_{\mathbf{G}}$ be the possibility distribution associated with $\mathbb{G}$. Then,*
$$\pi_{\mathbf{G}} \models_P \psi \text{ iff } \bigvee_{\phi_i \in \rho_\Delta(\mathbf{G})} \phi_i \vdash \psi.$$

2. *Let $\mathbb{G}$ be a set of goals and φ be a formula. Let $\mathbb{G}' = \{\mathcal{G}(\phi_i \wedge \varphi) = a_i : \mathcal{G}(\phi_i) = a_i \in \mathbb{G}\}$. Then,*
$$\pi \models_\varphi \psi \text{ iff } \bigvee_{\phi'_i \in \rho_\Delta(\mathbf{G}')} \phi'_i \vdash \psi.$$

6 Merging multiple agents preferences in a bipolar representation

This section discusses the problem of merging agents preferences from the semantic and the syntactic points of view. The result of the merging process will also be a pair $(\mathbb{R}_{\oplus_{\mathbf{R}}}, \mathbb{G}_{\oplus_{\mathbf{G}}})$, where $\mathbb{R}_{\oplus_{\mathbf{R}}}$ is the result of merging agents rejections, and $\mathbb{G}_{\oplus_{\mathbf{G}}}$ is the result of merging agents positive goals. These two merging steps are processed separately, using generally different merging operators. They are described in the next two subsections.

6.1 Fusion of negative desires

Let $\{\mathbb{R}_1, \cdots, \mathbb{R}_n\}$ be a set of rejection bases provided by n agents to be merged with some merging operator $\oplus_{\mathbf{R}}$. $\oplus_{\mathbf{R}}$ is a function from $[0,1]^n$ to $[0,1]$. Since rejections have an immediate encoding in terms of possibilistic knowledge bases, we can apply the merging procedures of possibilistic knowledge bases [4] for merging rejection bases. In particular, this allows to have the syntactic counterpart of any semantic merging operator, satisfying minimal properties. See [4] for details.

In this section, we define a class of operators which seem to be appropriate for merging rejections. The idea is that if some piece of information is rejected by some agent, then it should be explicitely rejected after the merging process. Such kind of behaviour is captured by conjunctive operators. Let ϕ be a classical formula. To see if a formula ϕ should be explicitely stated as rejected (or not) and with what degree of rejection in $\mathbb{R}_{\oplus_{\mathbf{R}}}$ (the result of merging), we compute from each $\mathbb{R}_i$ its level of rejection a_i. Then, ϕ will be explicitely stated in $\mathbb{R}_{\oplus_{\mathbf{R}}}$ with a degree $\oplus_{\mathbf{R}}(a_1, \cdots, a_n)$.

Natural properties of $\oplus_{\mathbf{R}}$ are:

i) $\oplus_{\mathbf{R}}(0, \cdots, 0) = 0$.
If a piece of information is not explicitely rejected by any agent then it should not be explicitely rejected in $\mathbb{R}_{\oplus_{\mathbf{R}}}$.

ii) If $\forall i = 1, \cdots, n$, $a_i \geq b_i$ then
$\oplus_{\mathbf{R}}(a_1, \cdots, a_n) \geq \oplus_{\mathbf{R}}(b_1, \cdots, b_n)$
(monotonicity property).

iii) $\oplus_{\mathbf{R}}(0, \cdots, 0, a, 0, \cdots, 0) = a$.
Namely, if a formula is only explicitely rejected by one agent then its level of rejection should not increase in the result of merging. In fact, it will be rejected with the same level.

Note that *i*), *ii*) and *iii*) imply:

$$\text{if } a_i > 0 \text{ for some } i \text{ then } \oplus_{\mathbf{R}}(a_1, \cdots, a_n) > 0,$$

which corresponds to say that if a formula is rejected by at least one agent then it should be explicitly rejected in the result of merging.
It is also easy to check that $\oplus_{\mathbf{R}}(a_1, \cdots, a_n) \geq max(a_1, \cdots, a_n)$. Indeed, from *ii*) and *iii*), we have $\oplus_{\mathbf{R}}(a_1, \cdots, a_n) \geq \oplus_{\mathbf{R}}(0, \cdots, 0, a_i, 0, \cdots, 0) = a_i$. Hence, $\oplus_{\mathbf{R}} = max$ represents the most cautious merging operator, in the sense that a formula is not rejected more than what is stated by the more exigent agent (who more rejects ϕ). In this case, we simply have (for two bases):

$$\mathbb{R}_{max} = \mathbb{R}_1 \cup \mathbb{R}_2.$$

Note that combining rejection strengths with the maximum operation leads to combine possibility distributions associated with agents rejections with the minimum operator, namely $\pi_{\mathbf{R}max} = min(\pi_{\mathbf{R}1}, \pi_{\mathbf{R}2})$.
Since strengths are associated with weights, it is also possible to express reinforcement. For instance, a formula which is weakly rejected by different agents can be strongly rejected in the merging result. The operator $\oplus_{\mathbf{R}}(a, b) = a + b - ab$, which corresponds to the product of the distributions, allows to go beyond $max(a, b)$ without reaching 1 (strong rejection) if a and b differ from 1 (i.e., it is not strongly rejected by any agent).

Example 3 *Let $\mathbb{R}_1$ be the base given in Example 1, namely $\mathbb{R}_1 = \{\mathcal{R}(be \wedge \neg su) = 1, \mathcal{R}(co \wedge su) = 1, \mathcal{R}(\neg be \wedge su) = \frac{1}{2}\}$. Let $\mathbb{R}_2 = \{\mathcal{R}(be \wedge \neg su) = 1, \mathcal{R}(co \wedge su) = 1, \mathcal{R}(mo) = \frac{1}{2}\}$, where "mo" stands for mountain. Let $\oplus_{\mathbf{R}} = max$. Then,*
$\mathbb{R}_{max} = \{\mathcal{R}(be \wedge \neg su) = 1, \mathcal{R}(co \wedge su) = 1,$
$\mathcal{R}(\neg be \wedge su) = \frac{1}{2}, \mathcal{R}(mo) = \frac{1}{2}\}$.

6.2 Fusion of positive desires

We now discuss the merging of positive goals. We first provide a general result on the syntactic fusion of goals, similar to the one in [4] for classical possibilistic knowledge bases.
Let $\mathbb{G}_1, \cdots, \mathbb{G}_m$ be m bases of goals and $\pi_{\mathbf{G}1}, \cdots, \pi_{\mathbf{G}m}$ be their associated possibility distributions given by Definition 6. Let $\oplus_{\mathbf{G}}$ be a merging operator satisfying the following requirements:

- $\oplus_{\mathbf{G}}(0, \cdots, 0) = 0$.
- If $\forall j = 1, \cdots, m$, $a_j \geq b_j$ then
$\oplus_{\mathbf{G}}(a_1, \cdots, a_m) \geq \oplus_{\mathbf{G}}(b_1, \cdots, b_m)$.

The first requirement expresses that if a solution is not satisfactory for any agent then it should not be satisfactory in the result of the merging. The second property is simply the monotonicity property.
Let us restrict ourselves, for the sake of simplicity, to the case of two bases. Then, the following proposition gives the positive goals base associated with $\oplus_{\mathbf{G}}(\pi_{\mathbf{G}1}, \pi_{\mathbf{G}2})$:

Proposition 3 *Let $\mathbb{G}_1 = \{\mathcal{G}(\phi_i) = a_i : i = 1, \cdots, n\}$ and $\mathbb{G}_2 = \{\mathcal{G}(\psi_j) = b_j : j = 1, \cdots, m\}$ be two bases of positive goals. Let $\pi_{\mathbf{G}1}$ and $\pi_{\mathbf{G}2}$ be their associated possibility distributions. Let $\oplus_{\mathbf{G}}$ be a merging operator. Then, the base of positive goals associated with $\oplus_{\mathbf{G}}(\pi_{\mathbf{G}1}, \pi_{\mathbf{G}2})$ is:*
$\mathbb{G}_{\oplus_{\mathbf{G}}} = \{\mathcal{G}(\phi_i) = \oplus_{\mathbf{G}}(a_i, 0) : \mathcal{G}(\phi_i) = a_i \in \mathbb{G}_1\}$
$\cup \{\mathcal{G}(\psi_j) = \oplus_{\mathbf{G}}(0, b_j) : \mathcal{G}(\psi_j) = b_j \in \mathbb{G}_2\}$
$\cup \{\mathcal{G}(\phi_i \wedge \psi_j) = \oplus_{\mathbf{G}}(a_i, b_j) :$
$\mathcal{G}(\phi_i) = a_i \in \mathbb{G}_1$ *and* $\mathcal{G}(\psi_j) = b_j \in \mathbb{G}_2\}$.

The choice of the merging operator for combining $\pi_{\mathbf{G}1}$ and $\pi_{\mathbf{G}2}$ is less contrained since several merging operators, can be considered, such as conjunctive, disjunctive and also "intermediary" operators which reinforce what is common and discount the goals which only concern one agent (see [14] for a representation of such operators).
If the agents are highly cooperative then we can say that an agent adds to its goals those of the other agent provided that they do not contradict what is acceptable for him. In this case $\oplus_{\mathbf{G}} = max$ is recommended.

Example 4 *Let $\mathbb{G}_1$ be the base given in Example 2, namely*
$\mathbb{G}_1 = \{\mathcal{G}(\neg co \wedge be \wedge su) = 1, \mathcal{G}(co \wedge \neg be \wedge \neg su) = \frac{1}{4}\}$.
Let $\mathbb{G}_2 = \{\mathcal{G}(co \wedge \neg be \wedge \neg su) = \frac{1}{2}, \mathcal{G}(\neg be \wedge \neg su) = \frac{1}{4}\}$.
Let $\mathbb{G}_{max}$ be the result of combining $\mathbb{G}_1$ and $\mathbb{G}_2$ with $\oplus_{\mathbf{G}} = max$. Then,
$\mathbb{G}_{max} = \{\mathcal{G}(\neg co \wedge be \wedge su) = 1, \mathcal{G}(co \wedge \neg be \wedge \neg su) = \frac{1}{4},$
$\cup \{\mathcal{G}(co \wedge \neg be \wedge \neg su) = \frac{1}{2}, \mathcal{G}(\neg be \wedge \neg su) = \frac{1}{4}\}$
$\cup \{\mathcal{G}(\bot) = 1, \mathcal{G}(co \wedge \neg be \wedge \neg su) = \frac{1}{2}\}$
which is semantically equivalent to
$\mathbb{G}_{max} = \{\mathcal{G}(\neg co \wedge be \wedge su) = 1,$
$\mathcal{G}(co \wedge \neg be \wedge \neg su) = \frac{1}{2}, \mathcal{G}(\neg be \wedge \neg su) = \frac{1}{4}\}$
since $\mathcal{G}(co \wedge \neg be \wedge \neg su) = \frac{1}{4}$ is subsumed by $\mathcal{G}(\neg be \wedge \neg su) = \frac{1}{4}$ (see Definition 7).

Note that this mode of merging corresponds to considering that what is satisfactory by one agent is also satisfactory by the other.

7 Coherence of positive and negative desires and how to restore it

Merging agents preferences can lead to conflicts. This section first defines the notion of coherence between rejections and positive goals, and shows how to restore the coherence in case of inconsistency.

Let $(\mathbb{R}, \mathbb{G})$ be preferences for an agent (or a group of agents). Intuitively, if $\mathbb{R} = \{\mathcal{R}(\phi_i) = 1 : i = 1, \cdots, n\}$ and $\mathbb{G} = \{\mathcal{G}(\psi_j) = 1 : j = 1, \cdots, m\}$ are classical logic bases (without rejection or satisfaction weights) then $\mathbb{R}$ and $\mathbb{G}$ are coherent if

$$\bigvee_{i=1,\cdots,m} \psi_j \vdash \bigwedge_{i=1,\cdots,n} \neg\phi_i,$$

namely any solution satisfying at least one goal of $\mathbb{G}$ should falsify all formulas in $\mathbb{R}$. More generally, any interpretation which is satisfactory to a degree a (w.r.t. $\mathbb{G}$) should be at least feasible to a degree a (w.r.t. $\mathbb{R}$).

Definition 10 *Let $\pi_{\mathbf{G}}$ and $\pi_{\mathbf{R}}$ be the two possibility distributions representing respectively the positive goals and rejections of an agent. Then, $\pi_{\mathbf{G}}$ and $\pi_{\mathbf{R}}$ are said to be coherent iff*

$$\forall\omega, \pi_{\mathbf{G}}(\omega) \leq \pi_{\mathbf{R}}(\omega).$$

The coherence checking can also be done syntactically using the bases $\mathbb{G}$ and $\mathbb{R}$.

Proposition 4 *Let $\mathbb{G}$ and $\mathbb{R}$ be respectively the sets of goals and rejections of an agent. Then, $\mathbb{G}$ and $\mathbb{R}$ are said to be coherent (in the sense of Definition 10) iff: $\forall a \geq 0$,*

$\bigvee_{\mathcal{G}(\psi_j)=a_j \in \mathbf{G}, a_j \geq a} \psi_j \vdash \bigwedge_{\mathcal{R}(\phi_i)=a_i \in \mathbf{R}, a_i > 1-a} \neg\phi_i$.

In the merging process, the consistency of each pair $(\mathbb{R}_i, \mathbb{G}_i)$ does not guarantee the coherence of the pair $(\mathbb{R}, \mathbb{G})$, where $\mathbb{R}$ (resp. $\mathbb{G}$) is the result of merging $\mathbb{R}_i$'s (resp. $\mathbb{G}_i$'s), for most of the operators $(\oplus_{\mathbf{R}}, \oplus_{\mathbf{G}})$.

Example 5 *Let $\mathbb{R}$ and $\mathbb{G}$ be respectively the sets of merging rejections and merging goals computed in Examples 3 and 4. We have*

$\mathbb{R} = \{\mathcal{R}(be \wedge \neg su) = 1, \mathcal{R}(co \wedge su) = 1,$
$\mathcal{R}(\neg be \wedge su) = \frac{1}{2}, \mathcal{R}(mo) = \frac{1}{2}\}$, *and*
$\mathbb{G} = \{\mathcal{G}(\neg co \wedge be \wedge su) = 1,$
$\mathcal{G}(co \wedge \neg be \wedge \neg su) = \frac{1}{2}, \mathcal{G}(\neg be \wedge \neg su) = \frac{1}{4}\}$.

At the semantic level, consider the interpretation $\omega_0 = su \wedge \neg co \wedge be \wedge mo$. Then, we can check that $\pi_{\mathbf{R}}(\omega_0) = \frac{1}{2}$ while $\pi_{\mathbf{G}}(\omega_0) = 1$.

Observe that $\pi_{\mathbf{G}}(\omega_0) \not\leq \pi_{\mathbf{R}}(\omega_0)$, then $\pi_{\mathbf{G}}$ and $\pi_{\mathbf{R}}$ are not coherent.

At the syntactic level, let $a = 1$. Then,

$\bigvee\{\psi_j : \mathcal{G}(\psi_j) = a_j \in \mathbb{G}, a_j \geq 1\} = \neg co \wedge be \wedge su$, *and*
$\bigwedge_{\mathbf{R}(\phi_i)=a_i \in \mathbf{R}, a_i > 0} \neg\phi_i = (\neg be \vee su) \wedge (\neg co \vee \neg su) \wedge (be \vee \neg su) \wedge \neg mo$.

We have $\neg co \wedge be \wedge su \not\vdash (\neg be \vee su) \wedge (\neg co \vee \neg su) \wedge (be \vee \neg su) \wedge \neg mo$. Then, $(\mathbb{R}, \mathbb{G})$ is not coherent.

When the coherence condition is not satisfied by the results of the merging, this means that the set of goals resulting from merging the goals of the agents is not compatible with what is acceptable by the agents. A way to restore the coherence in the sense of Definition 10 is to revise either the possibility distribution $\pi_{\mathbf{G}}$ or the possibility distribution $\pi_{\mathbf{R}}$. We choose to revise $\pi_{\mathbf{G}}$ since in practice it is more difficult to question $\pi_{\mathbf{R}}$ which expresses rejections. The revision of $\pi_{\mathbf{G}}$ in this case consists in decreasing the possibility degree of each interpretation in $\pi_{\mathbf{G}}$ to the possibility degree of this interpretation in $\pi_{\mathbf{R}}$. In other terms, restoring the coherence leads to revise $\pi_{\mathbf{G}}$ into $\pi_{\mathbf{G}rev}$ as follows:

$$\forall\omega, \pi_{\mathbf{G}rev}(\omega) = min(\pi_{\mathbf{G}}(\omega), \pi_{\mathbf{R}}(\omega)).$$

At the syntactic level, this leads to weaken goals and to decrease the level of satisfaction associated with goals in $\mathbb{G}$, in the following way: let $a_1, \cdots, a_n$ be the weights of $\mathbb{R}$ such that $a_1 > \cdots > a_n > 0$ i.e., $\mathbb{R} = \{\mathcal{R}(\phi_1) = a_1, \cdots, \mathcal{R}(\phi_n) = a_n\}$. This is always possible by gathering formulas of the same weight in a unique formula.

Let $\mathcal{G}(\psi) = b \in \mathbb{G}$, and let a_i be the minimal weight such that $1 - a_i < b$. Then, $\mathbb{G}_{rev}$ is obtained by replacing each $\mathcal{G}(\psi) = b$ in $\mathbb{G}$ by:

$\{\mathcal{G}(\neg\phi_1 \wedge \cdots \wedge \neg\phi_i \wedge \psi) = b\}$
$\cup\{\mathcal{G}(\neg\phi_1 \wedge \cdots \wedge \neg\phi_k \wedge \psi) = 1 - a_{k+1} : k = 1, \cdots, i-1\}$
$\cup\{\mathcal{G}(\psi) = 1 - a_1\}$.

Intuitively this means that each solution satisfying the goal ψ and at least the formulas $\neg\phi_1, \cdots, \neg\phi_i$ is considered as satisfactory to at least the degree b. However we decrease the satisfaction degree of solutions which satisfy ψ but falsify at least one formula from $\{\neg\phi_1, \cdots, \neg\phi_i\}$.

This approach leads to the following result:

Proposition 5 *Let $\mathbb{G}$ and $\mathbb{R}$ be the sets of goals and rejections respectively. Let $\pi_{\mathbf{G}}$ and $\pi_{\mathbf{R}}$ be their associated possibility distributions respectively. Let $\pi_{\mathbf{G}rev}$ be defined as follows: $\forall\omega, \pi_{\mathbf{G}rev}(\omega) = min(\pi_{\mathbf{G}}(\omega), \pi_{\mathbf{R}}(\omega))$. Let $\mathbb{G}_{rev}$ be the base constructed from $\mathbb{G}$ and $\mathbb{R}$ above. Then, $\pi_{\mathbf{G}rev}$ is the possibility distribution associated with $\mathbb{G}_{rev}$ following Definition 6.*

Example 6 *Let us consider again the possibility distributions $\pi_{\mathbf{R}}$ and $\pi_{\mathbf{G}}$ associated with $\mathbb{R}$ and $\mathbb{G}$ computed in Examples 3 and 4 respectively. We have*
$\mathbb{R} = \{\mathcal{R}(be \wedge \neg su) = 1, \mathcal{R}(co \wedge su) = 1,$
$\mathcal{R}(\neg be \wedge su) = \frac{1}{2}, \mathcal{R}(mo) = \frac{1}{2}\}$ *and*
$\mathbb{G} = \{\mathcal{G}(\neg co \wedge be \wedge su) = 1,$
$\mathcal{G}(co \wedge \neg be \wedge \neg su) = \frac{1}{2}, \mathcal{G}(\neg be \wedge \neg su) = \frac{1}{4}\}.$
Let $\pi_{\mathbf{G}rev} = min(\pi_{\mathbf{G}}, \pi_{\mathbf{R}})$.
The goals base associated with $\pi_{\mathbf{G}rev}$ *is obtained by weakening (when necessary) the goals of* $\mathbb{G}$. *For example the goal* $\mathcal{G}(\neg co \wedge be \wedge su) = 1$ *is replaced by two goals* $\mathcal{G}(\neg co \wedge be \wedge su \wedge \neg mo) = 1$ *and* $\mathcal{G}(\neg co \wedge be \wedge su) = \frac{1}{2}$ *because the situation* $\neg co \wedge be \wedge su \wedge \neg mo$ *(induced by* $\neg co \wedge be \wedge su$*) does not satisfy any formula in* $\mathbb{R}$, *while the situation* $\neg co \wedge be \wedge su \wedge mo$ *satisfies mo so it is decreased to the degree* $\frac{1}{2}$. *With a similar treatment of other goals we get:*
$\mathbb{G}_{rev} = \{\mathcal{G}(\neg co \wedge be \wedge su \wedge \neg mo) = 1, \mathcal{G}(\neg co \wedge be \wedge su) = \frac{1}{2}, \mathcal{G}(co \wedge \neg be \wedge \neg su) = \frac{1}{2}, \mathcal{G}(\neg be \wedge \neg su) = \frac{1}{4}\}.$

8 Conclusion

This paper has advocated a bipolar framework for representing preferences accurately under the form of two sets of formulas having different semantics, both of them being encoded in the framework of possibilty theory. The representation framework remains simple (although each of the two sets could be equivalently represented under a graphical form, or as a set of conditionals [2]). Besides, the proposed model remains qualitative since only the ordering between the satisfaction levels or the rejection levels is meaningful.
The idea of a bipolar representation framework could accommodate more quantitative frameworks, like penalty logic [9], as well as when the weights are thought as penalities (e.g., as in [22, 13]).

In the penalty logic framework, one can also distinguish between two penalty bases representing rejections and positive goals respectively. The weights (which are then integer values) associated to formulas in the rejection base express the price to pay if the rejection is not respected. While the weights associated with formulas in the base of goals express the bonus to win if the desires are satisfied. In both cases, the weight associated with a given solution is the sum of the weights of falsified (in case of rejection), or satisfied (for the goals) formulas.

Another quantitative framework is the evidence theory framework [19, 21]. Belief functions have already been used for representing rejections. One can also use the so-called "commonality" function Q for representing positive goals. Indeed, a commonality function plays a role similar to the one of the guaranteed possibility measure Δ. Particularly when focal elements are nested, then the commonality function simply coincides with a guaranteed possibility measure.

Such a bipolar representation is also of interest when representing knowledge rather than preferences, as is discussed in [11], where the negative parts correspond to what is known as being (more or less impossible), while the positive parts gather worlds wich are guaranteed to be feasible because they have been observed or reported. This is why integrity constraints which are pieces of knowledge of the first kind, can be added when necessary to the negative part of the representation of the preferences, leading to the specification of what is acceptable because it is not impossible by integrity constraints or because of the taste of the user. Beyond the interest of representing preferences of agents for fusing them, and restoring coherence when necessary, the inference machinery of the possibilistic logic framework (extended to formulas weighted in terms of the function Δ) enables us to reason about preferences.

In the long range, such a representation scheme for preferences should turn to be useful for elicitating user's wishes in information or recommender systems, and handling preferences conflict between agents. In the full paper, we will also provide a comparative study with recent related works on preferences, e.g. [22].

References

[1] F. Bacchus and A. J. Grove. Utility Independence in a Qualitative Decision Theory. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation, (KR'96)*, pages 542-552, 1996.

[2] S. Benferhat, D. Dubois, S. Kaci, and H. Prade. Bridging logical, comparative and graphical possibilistic representation frameworks. In *6th European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU'01), LNAI 2143*, pages 422-431, 2001.

[3] S. Benferhat, D. Dubois, and H. Prade. Towards a possibilistic logic handling of preferences. *Applied Intelligence*, 14(3):303-317, 2001.

[4] S. Benferhat, D. Dubois, H. Prade, and M. Williams. A practical approach to fusing and revising prioritized belief bases. In *Proc. of EPIA 99, LNAI n° 1695, Springer Verlag*, pages 222-236, 1999.

[5] C. Boutilier. Toward a Logic for Qualitative Decision Theory. In *Proceedings of the 4th Inter-*

national Conference on Principles of Knowledge Representation, (KR'94), pages 75–86, 1994.

[6] C. Boutilier, R.I. Brafman, H. Hoos, and D. Poole. Reasoning with conditional ceteris paribus preference statements. In *Proceedings of 15th Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 71–80, 1999.

[7] J.T. Cacioppo and G.G. Bernston. The affect system: Architecture and operating characteristics. *Current Directions in Psychological Science*, 8, 5:133–137, 1999.

[8] J.T. Cacioppo, W.L. Gardner, and G.G. Bernston. Beyond bipolar conceptualizations and measures: The case of attitudes and evaluative space. *Personality and Social Psychology Review*, 1, 1:3–25, 1997.

[9] F. Dupin de Saint-Cyr, J. Lang, and T. Schiex. Penalty logic and its link with dempster-shafer theory. In *Proc. of 10th International Conference on Uncertainty in Artificial Intelligence (UAI'94)*, pages 204–211, 1994.

[10] J. Doyle, Y. Shoham, and M. P. Wellman. A Logic of Relative Desire. In *Proceedings of the 6th International Symposium on Methodologies for Intelligent Systems (ISMIS'91)*, pages 16–31, 1991.

[11] D. Dubois, P. Hajek, and H. Prade. Knowledge-driven versus data-driven logics. *Journal of Logic, Language, and Information*, 9:65–89, 2000.

[12] D. Dubois, J. Lang, and H. Prade. Possibilistic logic. *In Handbook of Logic in Artificial Intelligence and Logic Programming*, 3, Oxford University Press:pages 439–513, 1994.

[13] D. Dubois and H. Prade. Toll sets and toll logic. In *FUZZY LOGIC, State of the Art, (édité par R. Lowen and M. Roubens)*, pages 169–177. Kluwer Academic Publishers, 1993.

[14] D. Dubois and H. Prade. Possibility theory and data fusion in poorly informed environments. In *Control Engineering Practice*, volume 2(5), pages 811–823, 1994.

[15] D. Dubois and H. Prade. Possibility theory: qualitative and quantitative aspects. *In Handbook of Defeasible Reasoning and Uncertainty Management Systems. (D. Gabbay, Ph. Smets, eds.), Vol. 1: Quantified Representations of Uncertainty and Imprecision*, (Ph. Smets, ed.) Kluwer, Dordrecht:169–226, 1998.

[16] M. Grabisch and C. Labreuche. The Sipos integral for the aggregation of interacting bipolar criteria. In *8th International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU'00), Madrid*, pages 395–409, 2000.

[17] C. Lafage. Représentation de préférences en logique: application à la décision de groupe. In *Thèse de doctorat. Université Paul Sabatier. Toulouse.*, 2001.

[18] J. Lang. Conditional desires and utilities - and alternative logical approach to qualitative decision theory. In *Proceedings of 12th European Conference on Artificial Intelligence (ECAI'96)*, pages 318–322, 1996.

[19] G. Shafer. A mathematical theory of evidence. In *New Jersey: Princeton University Press*, 1976.

[20] Y. Shoham. Reasoning aboutchange. In *MIT Press, MA*, 1988.

[21] P. Smets and R. Kennes. The transferable belief model. *Artificial Intelligence*, 66:191–234, 1994.

[22] L. van der Torre and E. Weydert. Parameters for utilitarian desires in a qualitative decision theory. *Applied Intelligence*, 14:285–301, 2001.

[23] M. P. Wellman, J. Doyle, and Y. Shoham. Preferential semantics for goals. In *Proceedings of the 9th Conference on Artificial Intelligence, (AAAI'91)*. 698-703, 1991.

Appendix : Proofs of technical results

Proof of Proposition 1

Let $\mathbb{R} = \{\mathcal{R}(\phi_i) = a_i : i = 1, \cdots, n\}$ and $\Sigma = \{(\neg\phi_i, a_i) : \mathcal{R}(\phi_i) = a_i \in \mathbb{R}\}$.

Following Definition 5, we have :

$\pi_{\mathbb{R}}(\omega) = 1 - max\{a_i : \omega \models \phi_i, \mathcal{R}(\phi_i) = a_i \in \mathbb{R}\}$,

with $max\{\emptyset\} = 0$

which is equivalent to

$\pi_{\mathbb{R}}(\omega) =$

$$\begin{cases} 1 & \text{if } \forall \mathcal{R}(\phi_i) = a_i \in \mathbb{R}, \omega \not\models \phi_i \\ 1 - max\{a_i : \mathcal{R}(\phi_i) = a_i \in \mathbb{R} \text{ and } \omega \not\models \neg\phi_i\} & \text{otherwise.} \end{cases}$$

$$= \begin{cases} 1 & \text{if } \forall (\neg\phi_i, a_i) \in \Sigma, \omega \models \neg\phi_i \\ 1 - max\{a_i : (\neg\phi_i, a_i) \in \Sigma \text{ and } \omega \not\models \neg\phi_i\} & \text{otherwise.} \end{cases}$$

$= \pi_\Sigma(\omega)$,

where π_Σ is the possibility distribution associated with Σ following Definition 4.

■

Proof of Lemma 1

The two bases $\mathbb{G}$ and $\mathbb{G}'$ have the same formulas except the formulas $\mathcal{G}(\psi_1) = b, \mathcal{G}(\psi_2) = b$ which are only in $\mathbb{G}$ and $\mathcal{G}(\psi_1 \vee \psi_2) = b$ which is in $\mathbb{G}'$.

Then, it is clear that when $\pi_{\mathbb{G}}(\omega) > b$ and $\pi_{\mathbb{G}'}(\omega) > b$ (resp. $\pi_{\mathbb{G}}(\omega) < b$ and $\pi_{\mathbb{G}'}(\omega) < b$) we have $\pi_{\mathbb{G}}(\omega) = \pi_{\mathbb{G}'}(\omega)$.

Let us now show that $\forall\omega, \pi_{\mathbb{G}}(\omega) = b$ iff $\pi_{\mathbb{G}'}(\omega) = b$.

1- Suppose that $\pi_{\mathbb{G}}(\omega) = b$ which means that ω falsifies all formulas in $\mathbb{G}$ with a weight strictly greater than b. Then, $\pi_{\mathbb{G}'}(\omega) \leq b$ since $\mathbb{G}$ and $\mathbb{G}'$ have the same formulas with a weight strictly greater than b.

We now distinguish two cases:

- $\omega \models \psi_1$ (or $\omega \models \psi_2$):

Then, ω also satisfies $\psi_1 \vee \psi_2$. Hence, $\pi_{\mathbb{G}'}(\omega) = b$.

- $\omega \not\models \psi_1$ and $\omega \not\models \psi_2$, (hence $\omega \not\models \psi_1 \vee \psi_2$):

Then, $\pi_{\mathbb{G}}(\omega) = b$ means that ω satisfies a formula in $\mathbb{G}$ with a weight equal to b, other than ψ_1 and ψ_2. This formula also belongs to $\mathbb{G}'$. Then, $\pi_{\mathbb{G}'}(\omega) = b$.

2- Suppose that $\pi_{\mathbb{G}'}(\omega) = b$ which means that ω falsifies all formulas in $\mathbb{G}'$ with a weight strictly greater than b. Then, $\pi_{\mathbb{G}}(\omega) \leq b$ since $\mathbb{G}$ and $\mathbb{G}'$ have the same formulas with a weight strictly greater than b.

We now distinguish two cases:

- $\omega \models \psi_1 \vee \psi_2$:

This means that ω satisfies ψ_1, or ψ_2 or both. $\mathcal{G}(\psi_1) = b$ and $\mathcal{G}(\psi_2) = b$ belong to $\mathbb{G}$. Then, $\pi_{\mathbb{G}}(\omega) = b$.

- $\omega \not\models \psi_1 \vee \psi_2$:

Then, ω falsifies both ψ_1 and ψ_2. So, $\pi_{\mathbb{G}'}(\omega) = b$ means that ω satisfies a formula in $\mathbb{G}'$ with a weight equal to b, other than $\psi_1 \vee \psi_2$. This formula also belongs to $\mathbb{G}$. Then, $\pi_{\mathbb{G}}(\omega) = b$.

■

Proof of Lemma 2

$\mathbb{G}$ and $\mathbb{G}'$ contain the same formulas except $\mathcal{G}(\phi) = a$ which belong to $\mathbb{G}$ and not to $\mathbb{G}'$.

$\mathcal{G}(\phi) = a$ is subsumed in $\mathbb{G}$ means that there exists a formula $\mathcal{G}(\psi) = b$ in $\mathbb{G}$ with $b \geq a$ and $\phi \vdash \psi$.

$\phi \vdash \psi$ means that $\forall\omega$, if $\omega \models \phi$, then $\omega \models \psi$. Hence, $\forall\omega$, if $\omega \not\models \psi$, then $\omega \not\models \phi$.

Then, if an interpretation ω satisfies ψ then $\pi_{\mathbb{G}}(\omega) \geq b$. So, $\mathcal{G}(\phi) = a$ is not considered in the computation of $\pi_{\mathbb{G}}(\omega)$.

Now, if ω falsifies ψ, then it also falsifies ϕ. Then, $\mathcal{G}(\phi) = a$ is again not considered in the computation of $\pi_{\mathbb{G}}(\omega)$.

■

Proof of Proposition 2

1- To show that $\pi_{\mathbb{G}} \models_P \psi$ iff $\bigvee_{\phi_i \in \rho_\Delta(\mathbb{G})} \phi_i \vdash \psi$, we will show that $core(\pi_{\mathbb{G}}) = [\bigvee_{\phi_i \in \rho_\Delta(\mathbb{G})} \phi_i]$.

- Let us first show that $core(\pi_{\mathbb{G}}) \subseteq [\bigvee_{\phi_i \in \rho_\Delta(\mathbb{G})} \phi_i \vdash \psi]$.

Suppose that there exists an interpretation ω s.t. $\omega \in core(\pi_{\mathbb{G}})$ and $\omega \notin [\bigvee_{\phi_i \in \rho_\Delta(\mathbb{G})} \phi_i]$.

By definition $\rho_\Delta(\mathbb{G})$ is composed of the highest consistent formulas in $\mathbb{G}$ whose weight is equal to $Cons(\mathbb{G})$.

$\omega \notin [\bigvee_{\phi_i \in \rho_\Delta(\mathbb{G})} \phi_i]$ means that ω falsifies all highest consistent formulas in $\mathbb{G}$. Then, $\pi_{\mathbb{G}}(\omega) < Cons(\mathbb{G})$.

Let $\omega' \in [\bigvee_{\phi_i \in \rho_\Delta(\mathbb{G})} \phi_i]$. Then, $\pi_{\mathbb{G}}(\omega') = Cons(\mathbb{G})$.

Hence, $\pi_{\mathbb{G}}(\omega) < \pi_{\mathbb{G}}(\omega')$. However, this contradicts that ω belongs to $core(\pi_{\mathbb{G}})$.

- We now show that $[\bigvee_{\phi_i \in \rho_\Delta(\mathbb{G})} \phi_i] \subseteq core(\pi_{\mathbb{G}})$.

Suppose that there exists an interpretation ω s.t. $\omega \in [\bigvee_{\phi_i \in \rho_\Delta(\mathbb{R})} \phi_i]$ and $\omega \notin core(\pi_{\mathbb{G}})$.

$\omega \in [\bigvee_{\phi_i \in \rho_\Delta(\mathbb{G})} \phi_i]$ means that $\pi_{\mathbb{G}}(\omega) = Cons(\mathbb{G})$.

Now, $\omega \notin core(\pi_{\mathbb{G}})$ means that $\exists\omega', \pi_{\mathbb{G}}(\omega') > \pi_{\mathbb{G}}(\omega)$.

Let $\mathcal{G}(\phi) = a$ be the highest formula satisfied by ω'. $\pi_{\mathbb{G}}(\omega') > \pi_{\mathbb{G}}(\omega)$ means that $a > Cons(\mathbb{G})$. However, this contradicts the fact that $Cons(\mathbb{G})$ is the highest weight whose corresponding formula in $\mathbb{G}$ is consistent.

2- We now show that $core_\varphi(\pi_{\mathbb{G}}) = [\bigvee_{\phi'_i \in \rho_\Delta(\mathbb{G}')} \phi'_i]$ with $\mathbb{G}' = \{\mathcal{G}(\phi_i \wedge \varphi) = a_i : \mathcal{G}(\phi_i) = a_i \in \mathbb{G}\}$.

- We first show that $core_\varphi(\pi_{\mathbb{G}}) \subseteq [\bigvee_{\phi'_i \in \rho_\Delta(\mathbb{G}')} \phi'_i]$.

Suppose that there exists an interpretation ω s.t. $\omega \in core_\varphi(\pi_{\mathbb{G}})$ and $\omega \notin [\bigvee_{\phi'_i \in \rho_\Delta(\mathbb{G}')} \phi'_i]$.

By definition, $core_\varphi(\pi_{\mathbb{G}})$ is the set of interpretations satisfying φ and having the highest possibility degree in $\pi_{\mathbb{G}}$.

$\omega \notin [\bigvee_{\phi'_i \in \rho_\Delta(\mathbb{G}')} \phi'_i]$ means that ω falsifies all formulas of $\rho_\Delta(\mathbb{G}')$.

Recall that $\rho_\Delta(\mathbb{G}')$ is composed of highest consistent formulas in $\mathbb{G}'$ whose weight is equal to $Cons(\mathbb{G}')$.

Then, $\pi_{\mathbb{G}'}(\omega) < Cons(\mathbb{G}')$ i.e. $\pi_{\mathbb{G}'}(\omega) = a_j$ with

$a_j < Cons(\mathbb{G}')$ and $\omega \models \phi_j \wedge \varphi$.
Let ω' be a model in $[\bigvee_{\phi'_i \in \rho_\Delta(\mathbb{G}')} \phi'_i]$. Then, $\pi_{\mathbf{G}'}(\omega') > \pi_{\mathbf{G}'}(\omega)$. This means that $\omega' \models \phi_k \wedge \varphi$ s.t. $\mathcal{G}(\phi_k \wedge \varphi) = a_k \in \mathbb{G}'$ and $a_k > a_j$. However this contradicts the hypothesis that $\omega \in core_\varphi(\pi_{\mathbf{G}})$ which means that ω has the highest possibility degree in $\pi_{\mathbf{G}}$ with $\omega \models \varphi$.
- We now show that $[\bigvee_{\phi'_i \in \rho_\Delta(\mathbb{G}')} \phi'_i] \subseteq core_\varphi(\pi_{\mathbf{G}})$.
Suppose that there exists an interpretation ω s.t. $\omega \in [\bigvee_{\phi'_i \in \rho_\Delta(\mathbb{G}')} \phi'_i]$ and $\omega \notin core_\varphi(\pi_{\mathbf{G}})$.
$\omega \in [\bigvee_{\phi'_i \in \rho_\Delta(\mathbb{G}')} \phi'_i]$ means that $\pi_{\mathbf{G}'}(\omega) = Cons(\mathbb{G}')$.
$\omega \notin core_\varphi(\pi_{\mathbf{G}})$ means that $\exists \omega', \omega' \models \varphi$ and $\pi_{\mathbf{G}}(\omega') > \pi_{\mathbf{G}}(\omega)$.
$\pi_{\mathbf{G}'}(\omega') > \pi_{\mathbf{G}'}(\omega)$ means that $\pi_{\mathbf{G}'}(\omega') > Cons(\mathbb{G}')$. However, this contradicts the hypothesis that $Cons(\mathbb{G}')$ corresponds to the weight of the highest consistent formula in $\mathbb{G}'$.
■

Proof of Proposition 3
Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two bases of positive goals and, $\pi_{\mathbf{G}1}$ and $\pi_{\mathbf{G}2}$ be their associated possibility distributions. Let $\oplus_{\mathbf{G}}$ be a merging operator, and $\mathbb{G}_{\oplus_{\mathbf{G}}}$ be the result of merging $\mathbb{G}_1$ and $\mathbb{G}_2$ using $\oplus_{\mathbf{G}}$ i.e.,
$\mathbb{G}_{\oplus_{\mathbf{G}}} = \{\mathcal{G}(\phi_i) = \oplus_{\mathbf{G}}(a_i, 0) : \mathcal{G}(\phi_i) = a_i \in \mathbb{G}_1\}$
$\cup \{\mathcal{G}(\psi_j) = \oplus_{\mathbf{G}}(0, b_j) : \mathcal{G}(\psi_j) = b_j \in \mathbb{G}_2\}$
$\cup \{\mathcal{G}(\phi_i \wedge \psi_j) = \oplus_{\mathbf{G}}(a_i, b_j) : \mathcal{G}(\phi_i) = a_i \in \mathbb{G}_1$ and $\mathcal{G}(\psi_j) = b_j \in \mathbb{G}_2\}$.
Let π be the possibility distribution associated with $\mathbb{G}_{\oplus_{\mathbf{G}}}$ using Definition 6 and let us show that $\pi = \oplus_{\mathbf{G}}(\pi_{\mathbf{G}1}, \pi_{\mathbf{G}2})$.
First, $\mathbb{G}_{\oplus_{\mathbf{G}}}$ can also be written in the following equivalent way:
$\{\mathcal{G}(\phi_i \wedge \psi_j) = \oplus_{\mathbf{G}}(a_i, b_j) : \mathcal{G}(\phi_i) = a_i \in \mathbb{G}'_1$ and $\mathcal{G}(\psi_j) = b_j \in \mathbb{G}'_2\}$,
with $\mathbb{G}'_1 = \mathbb{G}_1 \cup \{\mathcal{G}(\top) = 0\}$ and $\mathbb{G}'_2 = \mathbb{G}_2 \cup \{\mathcal{G}(\top) = 0\}$.
Indeed, we have gathered in this expression the three subsets of $\mathbb{G}_{\oplus_{\mathbf{G}}}$ into one set. The formulas $\mathcal{G}(\phi_i) = \oplus_{\mathbf{G}}(a_i, 0)$ are obtained by taking the conjunctions of $\mathcal{G}(\phi_i) = a_i$ and $\mathcal{G}(\top) = 0$.
Similarily the formulas $\mathcal{G}(\psi_j) = \oplus_{\mathbf{G}}(0, b_j)$ are obtained by taking the conjunctions of $\mathcal{G}(\psi_j) = b_j$ and $\mathcal{G}(\top) = 0$.
Note that we get moreover the formula $\mathcal{G}(\top) = \oplus_{\mathbf{G}}(0, 0)$ which however can be removed, since $\oplus_{\mathbf{G}}(0, 0) = 0$.
Then π is computed as follows:
$\pi(\omega) = max\{\oplus_{\mathbf{G}}(a_i, b_j) : \omega \models \phi_i \wedge \psi_j$ and $\mathcal{G}(\phi_i \wedge \psi_j) = \oplus_{\mathbf{G}}(a_i, b_j) \in \mathbb{G}_{\oplus_{\mathbf{G}}}\}$
$= max\{\oplus_{\mathbf{G}}(a_i, b_j) : \omega \models \phi_i \wedge \psi_j$ and $\mathcal{G}(\phi_i) = a_i \in \mathbb{G}'_1, \mathcal{G}(\psi_j) = b_j \in \mathbb{G}'_2\}$
$= max\{\oplus_{\mathbf{G}}(a_i, b_j) : \omega \models \phi_i, \mathcal{G}(\phi_i) = a_i \in \mathbb{G}'_1$ and $\omega \models \psi_j, \mathcal{G}(\psi_j) = b_j \in \mathbb{G}'_2\}$.
Since $\oplus_{\mathbf{G}}$ satisfies the monotonicity property then when a_i and b_j are maximal then $\oplus_{\mathbf{G}}(a_i, b_j)$ is also maximal.
Hence, $\pi(\omega) = \oplus_{\mathbf{G}}(max\{a_i : \omega \models \phi_i, \mathcal{G}(\phi_i) = a_i \in \mathbb{G}'_1, \}\ max\{b_j : \omega \models \psi_j, \mathcal{G}(\psi_j) = b_j \in \mathbb{G}'_2\}) = \oplus_{\mathbf{G}}(\pi_{\mathbf{G}'1}, \pi_{\mathbf{G}'2})$
$= \oplus_{\mathbf{G}}(\pi_{\mathbf{G}1}, \pi_{\mathbf{G}2})$ since $\pi_{\mathbf{G}'1} = \pi_{\mathbf{G}1}$ and $\pi_{\mathbf{G}'2} = \pi_{\mathbf{G}2}$ since $\mathcal{G}(\top) = 0$ can be removed from $\mathbb{G}'_1$ and $\mathbb{G}'_2$.
■

Proof of Proposition 4
The proof can be checked by noticing that Definition 10 is equivalent to
$\forall a \geq 0$, $\{\omega : \pi_{\mathbf{G}}(\omega) \geq a\} \subseteq \{\omega : \pi_{\mathbf{R}}(\omega) \geq a\}$,
and by noticing that
$[\bigvee_{\mathcal{G}(\psi_j) = a_j \in \mathbf{G}, a_j \geq a} \psi_j] = \bigcup_{\mathcal{G}(\psi_j) = a_j \in \mathbf{G}, a_j \geq a} [\psi_j]$
$= \{\omega : \pi_{\mathbf{G}}(\omega) \geq a\}$
and
$[\bigwedge_{\mathcal{R}(\phi_i) = a_i \in \mathbf{R}, a_i > 1-a} \neg\phi_i] = \bigcap_{\mathcal{R}(\phi_i) = a_i \in \mathbf{R}, a_i > 1-a} [\neg\phi_i]$
$= \{\omega : \pi_{\mathbf{R}}(\omega) \geq a\}$.
■

Proof of Proposition 5
Let $\mathbb{R} = \{\mathcal{R}(\phi_i) = a_i : i = 1, \cdots, n\}$ and
$\mathbb{G} = \{\mathcal{G}(\psi_j) = b_j : j = 1, \cdots, m\}$.
Let $\mathbb{G}_{rev}$ be the base of positive goals obtained from $\mathbb{G}$ by replacing each goal $\mathcal{G}(\psi) = b$ in $\mathbb{G}$ by the following goals:
$\{\mathcal{G}(\neg\phi_1 \wedge \cdots \wedge \neg\phi_i \wedge \psi) = b\}$
$\cup \{\mathcal{G}(\neg\phi_1 \wedge \cdots \wedge \neg\phi_k \wedge \psi) = 1 - a_{k+1} : k = 1, \cdots, i-1\}$
$\cup \{\mathcal{G}(\psi) = 1 - a_1\}$
where a_i be the minimal weight in $\mathbb{R}$ such that $1 - a_i < b$.
Let ω be an interpretation. Suppose that $\pi_{\mathbf{R}}(\omega) = 1 - a_i$ and $\pi_{\mathbf{G}}(\omega) = b_j$.
We distinguish two cases:

1. $b_j \leq 1 - a_i$:
   $\pi_{\mathbf{R}}(\omega) = 1 - a_i$ means that $\omega \models \neg\phi_1 \wedge \cdots \wedge \neg\phi_{i-1}$ and $\omega \not\models \neg\phi_1 \wedge \cdots \wedge \neg\phi_i$.
   Also $b_j \leq 1 - a_i$ means that if there is a weight a_k in $\mathbb{R}$ s.t. $1 - a_k < b_j$, then we have necessarily $1 - a_k < 1 - a_i$. Then, $a_k > a_i$. Hence $\omega \models \neg\phi_1 \wedge \cdots \wedge \neg\phi_k$.
   So the highest formula in $\mathbb{G}$ satisfied by ω is $\neg\phi_1 \wedge \cdots \wedge \neg\phi_k \wedge \psi_j$ whose weight is equal to b_j. Then, $\pi_{\mathbf{G}rev}(\omega) = min(\pi_{\mathbf{R}}(\omega), \pi_{\mathbf{G}}(\omega))$.

2. $b_j > 1 - a_i$:
   Let a_k be the minimal weight in $\mathbb{R}$ s.t. $b_j > 1 - a_k$. So we have $a_k \leq a_i$.
   $\pi_{\mathbf{R}}(\omega) = 1 - a_i$ means that $\omega \models \neg\phi_1 \wedge \cdots \wedge \neg\phi_{i-1}$ and $\omega \not\models \neg\phi_1 \wedge \cdots \wedge \neg\phi_i$.
   Then, the highest formula in $\mathbb{G}$ satisfied by ω is

$\mathcal{G}(\neg\phi_1 \wedge \cdots \wedge \neg\phi_{i-1} \wedge \psi_j) = 1 - a_i$. Hence,
$\pi_{\mathbf{Grev}}(\omega) = min(\pi_{\mathbf{R}}(\omega), \pi_{\mathbf{G}}(\omega))$.

■

Reasoning about Action II

Programming of Reasoning and Planning Agents with FLUX

Michael Thielscher*
Department of Computer Science
Dresden University of Technology
01062 Dresden (Germany)
mit@inf.tu-dresden.de

Abstract

We present a high-level programming method FLUX which allows to design cognitive agents that reason about their actions and plan. Based on the established, general action representation formalism of the Fluent Calculus, FLUX agents maintain an explicit, partial world model by which they control their own behavior. Thanks to the extensive reasoning facilities provided by the underlying calculus, FLUX allows to implement complex strategies with concise and modular agent programs. Systematic experiments with problems that require to reason about the performance of thousands of actions, have shown that FLUX exhibits excellent computational behavior and scales up particularly well to long-term control.

1 INTRODUCTION

One of the most challenging and promising goals of Artificial Intelligence research is the design of autonomous agents, including robots, that explore partially known environments and that are able to act sensibly under incomplete information. Autonomy in solving complex tasks requires the high-level cognitive capabilities of reasoning and planning: Exploring their environment, agents reason when they interpret sensor information, memorize it, and draw inferences from combined sensor data. Acting under incomplete information, agents employ their reasoning facilities to ensure that they are acting cautiously, and they plan ahead some of their actions with a specific goal in mind.

Agents whose intelligence goes beyond simple reactions to stimuli, reason and plan on the basis of a mental model of the state of their environment. As they move along, these agents constantly update this model to reflect the changes they have effected and the sensor information they have acquired. Having agents maintain an internal world model is necessary if we want them to choose their actions not only on the basis of the current status of their sensors but also by taking into account what they have previously observed or done. Moreover, the ability to reason about sensor information is necessary if properties of the environment can only indirectly be observed and require the agent to combine observations made at different stages. The cognitive capability of planning, finally, allows an agent to first calculate the effect of different action sequences in order to help it choosing one that is appropriate under the current circumstances.

Standard programming languages for agents, such as Java, require programmers to write special-purpose modules if they intend to endow their agents with the cognitive capabilities of reasoning and planning for the domain at hand. Formal theories of reasoning about actions and change, on the other hand, have the expressive power to provide these capabilities. Examples of existing agent programming methods deriving from general action theories are GOLOG [Levesque *et al.*, 1997; Reiter, 2001], based on the Situation Calculus, or the robot control language developed in [Shanahan and Witkowski, 2000], based on the Event Calculus. Neither of these systems and underlying calculi, however, provides the crucial concept of an explicit state representation. During the execution of a program, state knowledge is only indirectly represented via the initial conditions and the actions which the agent has performed thus far. As a consequence, evaluating conditions in an agent program always necessitates to trace back the entire history of actions, and hence requires ever increasing computational effort as the agent pro-

*Parts of the work reported in this paper have been carried out while the author was a visiting researcher at the University of New South Wales in Sydney, Australia.

gresses. Studies reported in an accompanying paper have shown that this concept fails to scale up to long-term agent control [Thielscher, 2002a].

An explicit state representation being a fundamental concept in the Fluent Calculus [Thielscher, 1999], this established and versatile action representation formalism [Thielscher, 2000a] offers an alternative theory as the formal underpinnings for a high-level agent programming method. Actions are specified in the Fluent Calculus by so-called state update axioms and knowledge update axioms, respectively, which can be readily used in agent programs for maintaining an internal world model in accordance with the performed actions and acquired sensor information. The Fluent Calculus is also equipped with the formal concept of action histories, so-called situations, which can be used by agents to solve planning tasks along their way.

In this paper, we present the high-level programming method FLUX (for: *Fluent Executor*) which allows the design of intelligent agents that reason and plan on the basis of the Fluent Calculus. Using the paradigm of constraint logic programming, FLUX comprises a method for encoding incomplete states along with a technique of updating these states according to a declarative specification of the elementary actions and sensing capabilities of an agent. With its powerful constraint solver, the underlying FLUX kernel provides general reasoning facilities, so that the agent programmer can focus on designing the high-level behavior. Allowing for concise programs and supporting modularity, our method promises to be eminently suitable for programming complex strategies for artificial agents. Moreover, systematic experiments have shown that FLUX exhibits excellent computational behavior and scales up particularly well to long-term control.

The rest of the paper is organized as follows. We begin with illustrating, in Section 2, the key features of our programming methodology by an example of a non-trivial agent program. In Section 3, the semantics of FLUX is given in terms of the Fluent Calculus. Section 4 contains a description of the FLUX kernel; a detailed account of the constraint solver and the proof of its correctness is given in an accompanying paper [Thielscher, 2002b]. In Section 5, we show how the Fluent Calculus allows to define and prove soundness of FLUX programs. In Section 6, we give an overview of studies showing the computational merits of FLUX. A brief outlook is given in Section 7.

The FLUX system, the example agent program, and the accompanying papers all are available for download at our web site `http://fluxagent.org`.

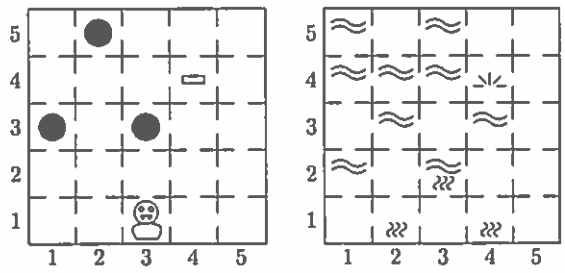


Figure 1: An example scenario in a Wumpus World where the 5×5-cave features three pits, the Wumpus in cell $(3,1)$, and gold in cell $(4,4)$. In the right hand side are depicted the corresponding perceptions (breeze, stench, glitter) for each location.

2 PROGRAMMING AGENTS IN FLUX

The example agent program in this paper is set in the artificial environment of the "Wumpus World," following the specification laid out in [Russell and Norvig, 1995] (see also Figure 1): An agent moves in a grid of cells, some of which contain, initially unknown to the agent, bottomless pits or gold, and there is one square which houses the hostile Wumpus. Its sensing capabilities allow the agent to perceive a breeze (a stench, respectively) if it is adjacent to a cell containing a pit (the Wumpus, respectively), and the agent notices a glitter in any cell containing gold and it hears a scream if the Wumpus gets killed. The elementary actions of the agent are to enter and exit the cave at cell $(1,1)$, turning clockwise by $90°$, going forward one square in the direction it faces, grabbing gold, and shooting an arrow in the direction it faces. The task of the agent is to find and bring home gold without falling into a pit and without encountering the Wumpus. A toy environment, the Wumpus World nonetheless requires crucial capabilities of intelligent agents, in particular acting cautiously under incomplete information, interpreting and logically combining sensor data, and planning. In the following, we develop a complete agent program for the Wumpus World, thereby illustrating the various features of our programming methodology.

2.1 FLUX States

FLUX agents use *states* as their internal model of the world. According to a convention in action theories, the atomic components of states are called *fluents*. Our program for the Wumpus World agent, for example, uses seven fluents: $At(x,y)$ and $Facing(d)$, representing that the agent is in cell (x,y) and faces

direction $d \in \{1,2,3,4\}$ (north, east, south, or west); $Gold(x,y)$, $Pit(x,y)$, and $Wumpus(x,y)$, representing that square (x,y) houses, respectively, gold, a pit, or the Wumpus; $Dead$, representing that the Wumpus is dead; and $Has(x)$, representing that the agent has $x \in \{Gold, Arrow\}$.

While a state is identified with all fluents that are true, agents hardly ever have complete information about their environment. To reflect this, incomplete states are encoded in FLUX as *open* lists, that is, lists with a variable tail, of fluents (possibly containing further variables). These lists are accompanied by *constraints* both for negated and disjunctive state knowledge as well as for variable range restrictions. The constraints are of the form $NotHolds(f,z)$, indicating that fluent f does not holds in state z; $NotHoldsAll(f,z)$, indicating that no instance of f holds in z; and $Or([f_1,\ldots,f_n],z)$, indicating that at least one of the fluents $f_1,\ldots,f_n$ holds in state z. Furthermore, FLUX employs a standard constraint solver for finite domains, which includes arithmetic constraints over rational numbers (using the equality and ordering predicates #=,#<,#> along with the standard functions +,-,*), range constraints (written `X::[a..b]`), and logical combinations using #/\ and #\/ for conjunction and disjunction, respectively. Consider, for example, the initial state of the Wumpus World agent, who has one arrow and knows that the Wumpus is not dead and could be in any square of the cave but $(1,1)$, that there are no pits in $(1,1)$ or outside the boundaries of the cave, and that initially the agent is nowhere inside of the cave and therefore not facing any direction:[1]

```
init(Z0) :- Z0 = [has(arrow),wumpus(WX,WY)|Z],
             [WX,WY] :: [1..5],
             not_holds(wumpus(1,1),Z0),
             not_holds_all(wumpus(_,_),Z),
             not_holds(dead,Z),
             not_holds(pit(1,1),Z),
             not_holds_all(pit(_,0),Z),  %boundary
             not_holds_all(pit(_,6),Z),
             not_holds_all(pit(0,_),Z),
             not_holds_all(pit(6,_),Z),
             not_holds_all(at(_,_),Z),   %agent
             not_holds_all(facing(_),Z),
             duplicate_free(Z0).
```

The reader may notice the difference in specifying the location of the Wumpus and the pits: While there is a unique but unknown cell housing the former, there can be many pits or none at all. Stipulating that pits may not lie outside the boundaries of the cave will simplify the specification of what it means to sense a breeze.

[1] The auxiliary constraint $DuplicateFree(z)$ stipulates that list z does not contain multiple occurrences.

2.2 Update Specifications

As agents move along, they need to update their internal world model whenever they perform an action, in order to reflect the changes that have been effected and the sensor information that has been acquired. This maintenance of the state is based on a specification of the elementary actions of the agent. Following the solution to the fundamental frame problem in the Fluent Calculus [Thielscher, 1999], FLUX uses so-called state update axioms, one for each action, defining the positive and negative effects and the meaning of sensing results. To this end, the FLUX kernel provides the predicate $Update(z_1,\vartheta^+,\vartheta^-,z_2)$, encoding that state z_2 is the result of updating state z_1 by the positive and negative effects ϑ^+ and ϑ^-, respectively. Both ϑ^+ and ϑ^- are finite, possibly empty lists of fluents. The auxiliary predicate $Holds(f,z)$, indicating that fluent f holds in state z, is also provided by the kernel to be used in state update axioms. On this basis, update axioms are encoded by defining the predicate $StateUpdate(z_1,a,z_2,\pi)$, defining state z_2 as the result of performing action a in state z_1 and perceiving sensor information π.

The update axioms used in our program for the Wumpus World agent, for example, are as follows. Entering the cave has the effect that the agent is in cell $(1,1)$ facing north. Furthermore, its sensors tell it whether it perceives a breeze, a stench, or a glitter. (The auxiliary predicates are defined below.):

```
state_update(Z1,enter,Z2,[B,S,G]) :-
  update(Z1,[at(1,1),facing(1)],[],Z2),
  breeze_perception(1,1,B,Z2),
  stench_perception(1,1,S,Z2),
  glitter_perception(1,1,G,Z2).
```

Exiting the cave has the reverse effect:

```
state_update(Z1,exit,Z2,[]) :-
  holds(facing(D),Z1),
  update(Z1,[],[at(1,1),facing(D)],Z2).
```

Turning has the effect of facing the next direction (clockwise) on the compass:

```
state_update(Z1,turn,Z2,[]) :-
  holds(facing(D),Z1),
  (D#<4 #/\ D1#=D+1) #\/ (D#=4 #/\ D1#=1),
  update(Z1,[facing(D1)],[facing(D)],Z2).
```

Going forward has the effect of being in the adjacent cell and acquiring new sensor inputs:

```
state_update(Z1,go,Z2,[B,S,G]) :-
  holds(at(X,Y),Z1), holds(facing(D),Z1),
  adjacent(X,Y,D,X1,Y1),
```

```
  update(Z1,[at(X1,Y1)],[at(X,Y)],Z2),
  breeze_perception(X1,Y1,B,Z2),
  stench_perception(X1,Y1,S,Z2),
  glitter_perception(X1,Y1,G,Z2).
```

Grabbing gold has the positive effect of having gold and the negative effect of clearing the cell:

```
state_update(Z1,grab,Z2,[]) :-
  holds(at(X,Y),Z1),
  update(Z1,[has(gold)],[gold(X,Y)],Z2).
```

Finally, shooting has a conditional effect, depending on whether the Wumpus is hit by the arrow, which is indicated by the perception of a scream. In either case the agent looses its arrow:

```
state_update(Z1,shoot,Z2,[S]) :-
  ( S=true, update(Z1,[dead],[has(arrow)],Z2)
    ; S=false, update(Z1,[],[has(arrow)],Z2) ).
```

Both the *Enter* action and the *Go* action use auxiliary predicates defining the meaning of perceiving a breeze, a stench, or a glitter at the new location:

```
breeze_perception(X,Y,Percept,Z) :-
  XE#=X+1, XW#=X-1, YN#=Y+1, YS#=Y-1,
  ( Percept=false, not_holds(pit(XE,Y),Z),
                   not_holds(pit(XW,Y),Z),
                   not_holds(pit(X,YN),Z),
                   not_holds(pit(X,YS),Z) ;
    Percept=true,
      or([pit(XE,Y),pit(X,YN),
          pit(XW,Y),pit(X,YS)],Z) ).
```

The clause for sensing a stench is identical but with *Pit* being replaced by *Wumpus*, whereas perceiving glitter indicates the presence of gold in the very cell:

```
glitter_perception(X,Y,Percept,Z) :-
  Percept=false, not_holds(gold(X,Y),Z) ;
  Percept=true,  holds(gold(X,Y),Z).
```

The update axiom for *Go* uses a further auxiliary predicate defining the notion of adjacent cells wrt. the different directions:

```
adjacent(X,Y,D,X1,Y1) :-
  [X,Y,X1,Y1]::1..5, D::1..4,
      (D#=1) #/\ (X1#=X)   #/\ (Y1#=Y+1) % north
  #\/ (D#=3) #/\ (X1#=X)   #/\ (Y1#=Y-1) % south
  #\/ (D#=2) #/\ (X1#=X+1) #/\ (Y1#=Y)   % east
  #\/ (D#=4) #/\ (X1#=X-1) #/\ (Y1#=Y).  % west
```

2.3 Agent Programs

Agent programs written in FLUX use the fundamental command $Execute(a, z_1, z_2)$. Resolving this predicate triggers the actual performance of action a. Furthermore, the update of the current state z_1 to state z_2 is inferred on the basis of the state update axiom for a. The expressive power of high-level agent programming becomes apparent when using the internal world model to control the continuation of a program. Conditioning in FLUX is based on the foundational predicates $Knows(f, z)$, $KnowsNot(f, z)$, and $KnowsVal(\vec{x}, f, z)$, representing that the agent knows that fluent f holds (respectively, does not hold) in state z, and that there exist ground instances of the variables in $\vec{x}$ such that fluent f is known to be true in state z. In the following we implement a simple strategy for a Wumpus World agent, allowing it to systematically and cautiously explore the cave. The program maintains three parameters: a list of choicepoints; a list of cells already visited; and the current path the agent has taken, which is used for backtracking.

To begin with, the agent enters the cave and sets the choicepoints for cell $(1, 1)$, i.e., north and east; the lists of visited cells and the backtrack path are initialized accordingly:

```
main :- init(Z0), execute(enter,Z0,Z1),
        Cpts=[1,1,[1,2]], Vis=[[1,1]], Btr=[],
        main_loop(Cpts,Vis,Btr,Z1).
```

In the main loop, the agent systematically selects a direction to explore from its current location. If this step is successful, then the agent tries to hunt the Wumpus and checks if gold is known to be in the current square; if so, it grabs the gold and goes home, otherwise choicepoints are created for the new location and both the list of visited nodes and the backtrack path are extended. If, on the other hand, the selected choicepoint cannot safely be explored, then it is removed; and if there are no choicepoints left for the current location, then the agent backtracks:

```
main_loop([X,Y,Choices|Cpts],Vis,Btr,Z) :-
  Choices=[Dir|Dirs] ->
    % choicepoint exists
    (explore(X,Y,Dir,Vis,Z,Z1) ->
       % successful choicepoint
       knows_val([X1,Y1],at(X1,Y1),Z1),
       hunt_wumpus(X1,Y1,Z1,Z2),
       (knows(gold(X1,Y1),Z2) ->
          execute(grab,Z2,Z3), go_home(Z3)
        ; Cpts1=[X1,Y1,[1,2,3,4],X,Y,Dirs|Cpts],
          Vis1=[[X1,Y1]|Vis], Btr1=[X,Y|Btr],
          main_loop(Cpts1,Vis1,Btr1,Z2) )
     % unsuccessful choicepoint
     ; main_loop([X,Y,Dirs|Cpts],Vis,Btr,Z) )
  % no choicepoints left
  ; backtrack(Cpts,Vis,Btr,Z).
```

Let us first consider the procedure for exploring a certain direction, which the agent does only if it has not

yet visited the adjacent square and if the new location is safe. Acting cautiously, the agent shall enter a cell only if the latter is known to be free of a pit and if either the Wumpus is known to be elsewhere or known to be dead:

```
explore(X,Y,D,V,Z1,Z2) :-
  adjacent(X,Y,D,X1,Y1), \+ member([X1,Y1],V),
  knows_not(pit(X1,Y1),Z1),
  (knows_not(wumpus(X1,Y1),Z1);knows(dead,Z1)),
  turn_to(D,Z1,Z), execute(go,Z,Z2).
turn_to(D,Z1,Z2) :-
  knows(facing(D),Z1) -> Z2=Z1
  ; execute(turn,Z1,Z), turn_to(D,Z,Z2).
```

Next, consider the procedure for backtracking, which simply means to go back to the first square in the list of backtracking points. If the list happens to be empty, then the agent has reached the home square and therefore exits the cave:

```
backtrack(_,_,[],Z) :- execute(exit,Z,_).
backtrack(Cpts,Vis,[X,Y|Btr],Z) :-
  go_back(X,Y,Z,Z1), main_loop(Cpts,Vis,Btr,Z1).
go_back(X,Y,Z1,Z2) :-
  holds(at(X1,Y1),Z1), adjacent(X1,Y1,D,X,Y),
  turn_to(D,Z1,Z), execute(go,Z,Z2).
```

A simple strategy for hunting the Wumpus, if it is still alive, is to check if the cell is known where it hides and if the agent happens to be in the same row or column. If so, then our agent turns into the direction of the Wumpus and shoots; otherwise, no action is performed and, hence, the state does not change:

```
hunt_wumpus(X,Y,Z1,Z2) :-
  \+ knows(dead,Z1),
  knows_val([WX,WY],wumpus(WX,WY),Z1),
  in_direction(X,Y,D,WX,WY)
  -> turn_to(D,Z1,Z), execute(shoot,Z,Z2)
   ; Z2=Z1.
in_direction(X,Y,D,X1,Y1) :-
  [X,Y,X1,Y1]::1..5, D::1..4,
      (D#=1) #/\ (X1#=X) #/\ (Y1#>Y)  % north
  #\/ (D#=3) #/\ (X1#=X) #/\ (Y1#<Y)  % south
  #\/ (D#=2) #/\ (X1#>X) #/\ (Y1#=Y)  % east
  #\/ (D#=4) #/\ (X1#<X) #/\ (Y1#=Y). % west
```

With just the procedure *GoHome* remaining to be defined, the FLUX program illustrates how high-level strategies for intelligent agents can be encoded in a concise and modular fashion. In particular, the reader may appreciate that there is no need to program any inference capabilities—these are fully provided by the underlying FLUX kernel. This allows the agent programmer to focus on specifying complex behaviors on the basis of the elementary actions of an agent.

Applied to the scenario depicted in Figure 1, the program runs as follows: After entering the cave the agent

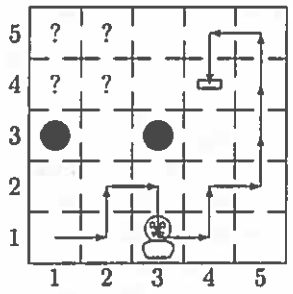


Figure 2: Exploring the cave depicted in Figure 1, our agent eventually reaches the cell with the gold, having shot the Wumpus along its way and inferred the locations of two pits. The northwest corner of the cave is still unknown territory.

first goes north to $(1,2)$. Since it senses a breeze and cannot decide whether there is a pit in $(1,3)$ or $(2,2)$, or both for that matter, the agent backtracks and then goes to $(2,1)$. Sensing no breeze there, the agent concludes that $(2,2)$ does not house a pit and that the breeze in $(1,2)$ must have come from $(1,3)$. Moreover, sensing a stench in $(2,1)$ without having experienced one in $(1,2)$, the agent infers that the Wumpus is in $(3,1)$. The agent shoots its arrow in that direction, making it possible to step over the dead Wumpus later on, and continues to explore the cave. Eventually, the agent arrives at $(4,4)$, where it senses a glitter and grabs the gold. At this stage, the current backtracking path is as depicted in Figure 2. Furthermore, the agent has acquired knowledge about the contents of all but four cells.

2.4 Planning

The ability to devise plans adds a second dimension to high-level agent programming. A useful application of planning in our example domain is to have the agent find a direct route home after claiming gold: While the agent could just follow its backtracking path, this usually involves a considerable detour, as the situation in Figure 2 illustrates.

Since unrestricted planning with incomplete states is a notoriously hard problem, FLUX allows to define compound actions and to specify domain-dependent search trees for planning problems. For instance, to plan a short but safe route home, our Wumpus World agent uses the compound action of going to any adjacent cell, thereby postponing to execution time the task of finding the right number of *Turn* actions. The virtual action of going to square (x,y) has the effect of changing the agent's location:

```
state_update(Z1,go_to(X,Y),Z2,[]) :-
  holds(at(X1,Y1),Z1),
  update(Z1,[at(X,Y)],[at(X1,Y1)],Z2).
```

Search trees for planning problems are specified in FLUX by defining the predicate $PlanProc(g,p)$, where g denotes a planning task and p describes the search space. Adopting key notions and notations from GOLOG [Levesque *et al.*, 1997], the concept of a search space in FLUX is defined as follows. An elementary action, a compound action, and a test $?(\varphi)$ are search spaces, and if $e_1, e_2, \ldots, e_n$ are search spaces, then so are $e_1 \# e_2$ (nondeterministic choice) and $[e_1, \ldots, e_n]$ (sequencing; $n \geq 0$). The search space for our example planning problem can thus be specified as follows. (The auxiliary tests are defined below.):

```
plan_proc(find_path(Vis),P) :-
  P = (?(home) #
       [?(poss_go(X,Y,Vis,Vis1)), go_to(X,Y),
        find_path(Vis1)]).
```

Put in words, a successful plan is either to be home or to go to an adjacent cell, if possible, followed by a plan to go home from there. The planning procedure employs a list of visited nodes to avoid running into a loop.

The tests $?(\varphi)$ occurring in the specification of a search space need to be accompanied by definitions for φ to hold in the current situation, that is, after performing the current sequence of actions s in the initial state z_0 of the planning problem. To this end, a clause with head $P(\vec{x}, z, s_0)$ needs to be defined for every test $?(P(\vec{x}))$. The FLUX kernel provides definitions of the standard predicates $Knows(f,s,z_0)$, $KnowsNot(f,s,z_0)$, and $KnowsVal(\vec{x},f,s,z_0)$, which carry the additional situation argument, too. On this basis, the tests needed for our example planning problem can be defined as follows:

```
home(S,Z) :-
  knows(at(1,1),S,Z).
poss_go(X1,Y1,Vis,Vis1,S,Z) :-
  knows_val([X,Y],at(X,Y),S,Z),
  ( D=1 ; D=2 ; D=3 ; D=4 ),
  adjacent(X,Y,D,X1,Y1),
  \+ member([X1,Y1],Vis),
  knows_not(pit(X1,Y1),Z),
  ( \+ knows(dead,Z)->knows_not(wumpus(X1,Y1),Z)
    ; true ),
  Vis1=[[X,Y]|Vis].
```

The reader may notice how the agent plans it safe, that is, only those locations are searched which are known to be free of a pit and of the alive wumpus. For the sake of efficiency, these tests refer to the initial state of the planning problem as they are not affected by the planned actions.

What remains to be done is to define the cost c of a plan p for a planning problem g, using the standard FLUX predicate $PlanCost(g,p,c)$. Furthermore, the execution of a virtual action a leading from state z_1 to state z_2 is to be defined using the standard FLUX predicate $ExecuteCompoundAction(a,z_1,z_2)$. The cost of a plan to go home is simply its length, while its execution requires to find the right number of turns prior to going to the adjacent cell:

```
plan_cost(find_path(_),P,C) :-
  length(P,C).
execute_compound_action(go_to(X,Y),Z1,Z2) :-
  holds(at(X1,Y1),Z1), adjacent(X1,Y1,D,X,Y),
  turn_to(D,Z1,Z), execute(go,Z,Z2).
```

Employing the FLUX predicates $Plan(g,p,z_1)$ and $Execute(p,z_1,z_2)$, agents can be programmed so as to find a plan p for problem g in state z_1 and to actually execute this plan, thereby updating state z_1 to z_2. In our example,

```
go_home(Z) :-
  plan(find_path([]),Plan,Z),
  execute(Plan,Z,Z1), execute(exit,Z1,_).
```

Applied to the state depicted in Figure 2, the agent finds and executes the plan to go straight down to $(4,1)$, to turn, and to walk straight to $(1,1)$.

3 A FLUENT CALCULUS SEMANTICS FOR FLUX

3.1 Fluents and States

The Fluent Calculus is a many-sorted predicate logic language with four standard sorts: FLUENT, STATE, ACTION, and SIT (for situations). States are composed of fluents (as atomic states) using the standard function $\circ$: STATE $\times$ STATE $\mapsto$ STATE and constant $\emptyset$: STATE (denoting the empty state). In order to capture the intuition of identifying a state with the fluents that hold, the special connection function of the Fluent Calculus should obey certain properties which resemble the union operation for sets:[2]

Definition 1 The foundational axioms Σ_{state} of the Fluent Calculus are,

1. Associativity, commutativity, idempotence, and

[2]Free variables in formulas are assumed universally quantified. Variables of sorts FLUENT, STATE, ACTION, and SIT shall be denoted by the letters f, z, a, and s, respectively. The function $\circ$ is written in infix notation.

unit element:

$$\begin{aligned}(z_1 \circ z_2) \circ z_3 &= z_1 \circ (z_2 \circ z_3)\\ z_1 \circ z_2 &= z_2 \circ z_1\\ z \circ z &= z\\ z \circ \emptyset &= z\end{aligned}$$

2. Empty state axiom:

$$\neg Holds(f, \emptyset)$$

3. Irreducibility and decomposition:

$$\begin{aligned}Holds(f_1, f_2) &\supset f_1 = f_2\\ Holds(f, z_1 \circ z_2) &\supset Holds(f, z_1) \vee Holds(f, z_2)\end{aligned}$$

4. State equality and state existence:

$$\begin{aligned}&[Holds(f, z_1) \equiv Holds(f, z_2)] \supset z_1 = z_2\\ &(\forall \Phi)(\exists z)(\forall f)\,(Holds(f, z) \equiv \Phi(f))\end{aligned}$$

where Φ is a second-order predicate variable of sort FLUENT while the macro *Holds* means that a fluent holds in a state:

$$Holds(f, z) \stackrel{\text{def}}{=} (\exists z')\, z = f \circ z' \qquad (1)$$

□

The very last, second-order axiom above stipulates the existence of a state for all possible combinations of fluents.

On this basis, the semantics of a state specification in FLUX of the form

$$z = [f_1, \ldots, f_k \mid z'],\ DuplicateFree(z)$$

is given by the equational axiom $z = f_1 \circ \ldots \circ f_k \circ z'$, where z, z' are of sort STATE and the f_i's are of sort FLUENT. The meaning of the FLUX atom $Holds(f, z)$ is as in macro (1), while the semantics for the constraints $NotHolds(f, z)$, $NotHoldsAll(f, z)$, and $Or([f_1, \ldots, f_n], z)$ is given by the following axioms:

$$\neg Holds(f, z);\quad (\forall \vec{x})\, \neg Holds(f, z);\quad \bigvee_{i=1}^{n} Holds(f_i, z)$$

where $\vec{x}$ are the free variables in f. For example, the initial state in the Wumpus World is suitably described by the following Fluent Calculus axiom (c.f. Section 2.1):

$$\begin{aligned}&(\exists x, y, z)\\ &\quad (\, State(S_0) = Has(Arrow) \circ Wumpus(x, y) \circ z \wedge\\ &\qquad 1 \le x \le 5 \wedge 1 \le y \le 5 \wedge\\ &\qquad \neg Holds(Wumpus(1,1), z_0) \wedge\\ &\qquad (\forall x', y')\, \neg Holds(Wumpus(x', y'), z) \wedge\\ &\qquad \neg Holds(Dead, z) \wedge \neg Holds(Pit(1,1), z) \wedge\\ &\qquad (\forall x')\, (\neg Holds(Pit(x', 0), z) \wedge\\ &\qquad\qquad \neg Holds(Pit(x', 6), z)\,) \wedge\\ &\qquad (\forall y')\, (\neg Holds(Pit(0, y'), z) \wedge\\ &\qquad\qquad \neg Holds(Pit(6, y'), z)\,) \wedge\\ &\qquad (\forall x', y')\, \neg Holds(At(x', y'), z) \wedge\\ &\qquad (\forall d)\, \neg Holds(Facing(d), z)\,)\end{aligned} \qquad (2)$$

The axiomatization of states in the Fluent Calculus via the foundational axioms paves the way for an extensional definition of addition and removal of (finitely many) fluents from states, which in turn lays the foundation for an effective solution to the fundamental frame problem in the presence of incomplete states. The following definition introduces the macro equation $z_1 - \tau = z_2$ with the intended meaning that state z_2 is state z_1 minus the fluents in the finite state τ. The compound macro $z_2 = (z_1 - \vartheta^-) + \vartheta^+$ means that state z_2 is state z_1 minus the fluents in ϑ^- plus the fluents in ϑ^+:

$$\begin{aligned}&z_1 - \emptyset = z_2 \stackrel{\text{def}}{=} z_2 = z_1\\ &z_1 - f = z_2\\ &\quad \stackrel{\text{def}}{=} (z_2 = z_1 \vee z_2 \circ f = z_1) \wedge \neg Holds(f, z_2)\\ &z_1 - (f_1 \circ f_2 \circ \ldots \circ f_n) = z_2\\ &\quad \stackrel{\text{def}}{=} (\exists z)\, (z = z_1 - f \wedge z_2 = z - (f_2 \circ \ldots \circ f_n))\\ &(z_1 - \vartheta^-) + \vartheta^+ = z_2\\ &\quad \stackrel{\text{def}}{=} (\exists z)\, (z = z_1 - \vartheta^- \wedge z_2 = z \circ \vartheta^+)\end{aligned}$$

where both ϑ^+, ϑ^- are finitely many FLUENT terms connected by "$\circ$". The crucial item is the second one, which defines removal of a single fluent f using a case distinction: Either $z_1 - f$ equals z_1 (which applies in case $\neg Holds(f, z_1)$), or $z_1 - f$ plus f equals z_1 (which applies in case $Holds(f, z_1)$). On this basis, the semantics of the FLUX predicate $Update(z_1, \vartheta^+, \vartheta^-, z_2)$ is given by the equation $z_2 = (z_1 - \vartheta^-) + \vartheta^+$.

3.2 Actions and Situations

Adopted from the Situation Calculus, the two standard sorts ACTION and SIT are used to axiomatize sequences of actions. The standard function Do : ACTION × SIT ↦ SIT denotes the situation reached by performing an action in a situation, and the constant S_0 : SIT denotes the initial situation. The standard function $State$: SIT ↦ STATE, which features uniquely in the Fluent Calculus, serves as denotation for the state of the environment in a situation.

Generalizing previous approaches [Bibel, 1986; Hölldobler and Schneeberger, 1990], the fundamental frame problem is solved in the Fluent Calculus by axioms which specify the difference between the states before and after an action [Thielscher, 1999]. Let $Poss$: ACTION × STATE denote that an action is possible in a state, then a *precondition axiom* for an action $A(\vec{x})$ is of the form

$$Poss(A(\vec{x}), z) \equiv \Pi(z)$$

where $\Pi(z)$ is a first-order formula with free variable z. The following is the general form of a *state update axiom* for a (possibly nondeterministic) action $A(\vec{x})$ with possibly conditional effects:[3]

$$\begin{array}{l} Poss(A(\vec{x}), s) \supset \\ \quad (\exists)\,(\,\Delta_1(s) \wedge \\ \qquad State(Do(A(\vec{x}), s)) = (State(s) - \vartheta_1^-) + \vartheta_1^+\,) \\ \quad \vee \ldots \vee \\ \quad (\exists)\,(\,\Delta_n(s) \wedge \\ \qquad State(Do(A(\vec{x}), s)) = (State(s) - \vartheta_n^-) + \vartheta_n^+\,) \end{array}$$

First-order formulas $\Delta_i(s)$ specify the conditions on $State(s)$ under which $A(\vec{x})$ has the positive and negative effects ϑ_i^+ and ϑ_i^-, respectively. Both ϑ_i^+ and ϑ_i^- are STATE terms consisting of FLUENTs only $(1 \leq i \leq n;\ n \geq 1)$.

Regarding our Wumpus World agent, consider the following precondition axioms:

$$\begin{array}{rcl} Poss(Enter, z) & \equiv & (\forall x, y)\,\neg Holds(At(x, y), z) \\ Poss(Exit, z) & \equiv & Holds(At(1, 1), z) \\ Poss(Turn, z) & \equiv & (\exists x, y)\,Holds(At(x, y), z) \\ Poss(Go, z) & \equiv & (\exists d, x, y, x', y') \\ & & (Holds(At(x, y), z) \wedge Holds(Facing(d), z) \wedge \\ & & Adjacent(x, y, d, x', y') \wedge \neg Holds(Pit(x, y), z) \wedge \\ & & [\neg Holds(Wumpus(x, y), z) \vee Holds(Dead, z)]\,) \\ Poss(Grab, z) & \equiv & (\exists x, y)\,(Holds(At(x, y), z) \wedge \\ & & \qquad Holds(Gold(x, y), z)) \\ Poss(Shoot, z) & \equiv & Holds(Has(Arrow), z) \end{array}$$

where $Adjacent(x, y, d, x', y')$ shall be defined as in Section 2.2.

The state update axioms are straightforward, following the description in Section 2.2. For example, the effect of Go on the state can be axiomatized as follows:

$$\begin{array}{l} Poss(Go, s) \supset \\ \quad (\exists d, x, y, x', y')\,(\,Holds(At(x, y), State(s)) \wedge \\ Holds(Facing(d), State(s)) \wedge Adjacent(x, y, d, x', y') \wedge \\ State(Do(Go, s)) = (State(s) - At(x, y)) + At(x', y')) \end{array}$$

The reader may notice that sensor information is not incorporated in state update axioms as sensing does not affect the state itself.

[3] Below, $Poss(a, s) \stackrel{\text{def}}{=} Poss(a, State(s))$.

3.3 Knowledge and Sensing

The basic Fluent Calculus has been extended in [Thielscher, 2000b] by the foundational predicate $KState$: SIT × STATE to allow for both representing state knowledge and reasoning about actions which involve sensing. An instance $KState(s, z)$ means that, according to the knowledge of the agent, z is a possible state in situation s. For example, the initial knowledge of our Wumpus World agent can be specified by this axiom:

$$(\forall z_0)\,(\,KState(S_0, z_0) \equiv \Psi(z_0)\,) \qquad (3)$$

where $\Psi(z_0)$ is formula (2) with $State(S_0)$ replaced by variable z_0. That is to say, all states which satisfy the initial specification are to be considered possible by the agent. In particular, the agent has no further prior knowledge of the cave.

Based on the notion of a knowledge state, a fluent is known to hold in a situation (not to hold, respectively) just in case it is true (false, respectively) in all possible states:

$$\begin{array}{rcl} Knows(f, s) & \stackrel{\text{def}}{=} & (\forall z)\,(KState(s, z) \supset Holds(f, z)) \\ Knows(\neg f, s) & \stackrel{\text{def}}{=} & (\forall z)\,(KState(s, z) \supset \neg Holds(f, z)) \end{array}$$

Moreover, a value of a fluent is known just in case a particular instance holds in all possible states:

$$\begin{array}{l} KnowsVal(\vec{x}, f, s) \stackrel{\text{def}}{=} \\ \quad (\exists \vec{x})(\forall z)(KState(s, z) \supset Holds(f, z)) \end{array}$$

For example, the axiomatization of the initial knowledge entails that the agent knows that the Wumpus must be somewhere,

$$\begin{array}{l} \Sigma_{state} \cup \{(3)\} \models \\ \quad KState(S_0, z_0) \supset (\exists x, y)\,Holds(Wumpus(x, y), z_0) \end{array}$$

but the agent does not know where,

$$\begin{array}{l} \Sigma_{state} \cup \{(3)\} \models \\ \quad \neg KnowsVal(x, y, Wumpus(x, y), S_0) \end{array}$$

The frame problem for knowledge is solved by axioms that determine the relation between the possible states before and after an action. More formally, the effect of an action $A(\vec{x})$, be it sensing or not, on the knowledge of the agent is specified by a so-called knowledge update axiom,[4]

$$\begin{array}{l} Knows(Poss(A(\vec{x})), s) \supset \\ \quad (KState(Do(A(\vec{x}), s), z) \equiv \\ \qquad (\exists z_1)(KState(s, z_1) \wedge \Psi(z, z_1) \wedge \Pi(z, s))\,) \end{array} \qquad (4)$$

[4] Below, macro $Knows(Poss(a), s)$ stands for the formula $(\forall z)\,(KState(s, z) \supset Poss(a, z))$.

where Ψ specifies the physical state update while Π restricts the possible states so as to agree with the actual state $State(s)$ on the sensed properties. For example, this is the knowledge update axiom for action Go of our Wumpus World agent:

$$\begin{array}{l} Knows(Poss(Go), s) \supset (KState(Do(Go, s), z) \equiv \\ \quad (\exists d, x, y, x', y', z_1)\,(KState(s, z_1) \wedge \\ \qquad Holds(At(x, y), z_1) \wedge Holds(Facing(d), z_1) \wedge \\ \qquad Adjacent(x, y, d, x', y') \wedge \\ \qquad z = (z_1 - At(x, y)) + At(x', y') \wedge \\ \qquad [Breeze(x', y', z) \equiv Breeze(x', y', State(s))] \wedge \\ \qquad [Stench(x', y', z) \equiv Stench(x', y', State(s))] \wedge \\ \qquad [Glitter(x', y', z) \equiv Glitter(x', y', State(s))])) \end{array}$$

where

$$\begin{array}{l} Breeze(x, y, z) \equiv \\ \quad Holds(Pit(x + 1, y), z) \vee Holds(Pit(x, y + 1), z) \vee \\ \quad Holds(Pit(x - 1, y), z) \vee Holds(Pit(x, y - 1), z) \end{array}$$

Likewise for $Stench(x, y, z)$, while

$$Glitter(x, y, z) \equiv Holds(Gold(x, y), z)$$

The other knowledge update axioms are straightforward and correspond to the encoding in Section 2.2.

4 THE FLUX KERNEL

Figure 3 gives an overview of the architecture of FLUX programs: The basic statements in agent programs are testing knowledge (predicate *Knows* etc.) and performing actions (predicate *Execute*), both of which are defined in the FLUX kernel. Maintaining the state when performing an action relies on the specification of update axioms, which in turn use the basic FLUX predicate *Update*. FLUX itself appeals to the paradigm of constraint logic programming, which enhances logic programs by mechanisms for solving constraints. In particular, so-called Constraint Handling Rules [Frühwirth, 1998] (CHRs) support declarative specifications of rules for processing the FLUX constraints which express negative and disjunctive state knowledge. In turn, these rules use finite domain constraints for handling variable arguments of fluents, which can be natural or rational numbers or of any user-defined finite domain.

4.1 States and Update

The basic definitions for states and update in the FLUX kernel are as follows:

```
holds(F,[F|_]).
holds(F,Z) :-
  nonvar(Z), Z=[F1|Z1], \+F==F1, holds(F,Z1).
holds(F,[F|Z],Z).
holds(F,Z,[F1|Zp]) :-
  nonvar(Z), Z=[F1|Z1], \+F==F1, holds(F,Z1,Zp).
minus(Z,[],Z).
minus(Z,[F|Fs],Zp) :-
  (knows_not(F,Z)->Z1=Z; holds(F,Z,Z1)),
  minus(Z1,Fs,Zp).
plus(Z,[],Z).
plus(Z,[F|Fs],Zp) :-
  (knows_not(F,Z)->Z1=[F|Z]; holds(F,Z), Z1=Z),
  plus(Z1,Fs,Zp).
update(Z1,P,N,Z2) :- minus(Z1,N,Z),plus(Z,P,Z2).
```

The accompanying paper [Thielscher, 2002b] contains a proof of correctness of these clauses wrt. the foundational axioms Σ_{state} of the Fluent Calculus and the axiomatic characterization of fluent removal and addition.

Knowledge in FLUX is identified with logical entailment wrt. incomplete state specifications, employing the principle of negation-as-failure:

```
knows(F,Z)       :- \+ not_holds(F,Z).
knows_not(F,Z)   :- \+ holds(F,Z).
knows_val(X,F,Z) :- holds(F,Z), \+ nonground(X).
```

Again we refer to [Thielscher, 2002b] for the formal correctness of this definition wrt. the theory of knowledge in the Fluent Calculus as outlined in Section 3.3.

Finally, action execution is defined as follows, whereby the predicate $Perform(a, \pi)$ is assumed to trigger the actual performance of elementary action a with sensing values π returned:

```
execute(E,Z1,Z2) :-
  E=[]     -> Z2=Z1 ;
  E=[A|P] -> execute(P,Z1,Z), execute(A,Z,Z2) ;
  elementary_action(E) ->
    perform(E,SV), state_update(Z1,E,Z2,SV) ;
  execute_compound_action(E,Z1,Z2).
```

The FLUX constraint solver consists in a small set of progression and evaluation CHRs dealing with the constraints for negative and disjunctive state knowledge. By these rules, constraints are constantly simplified and combined in the course of a program in order to draw new inferences and to detect inconsistencies. See [Thielscher, 2002b] for the complete set of CHRs. Thanks to their declarative nature, correctness of these rules is easily verified against the foundational axioms of the Fluent Calculus.

4.2 Planning

Planning in FLUX is based on searching for a situation, represented by a list of actions, which matches

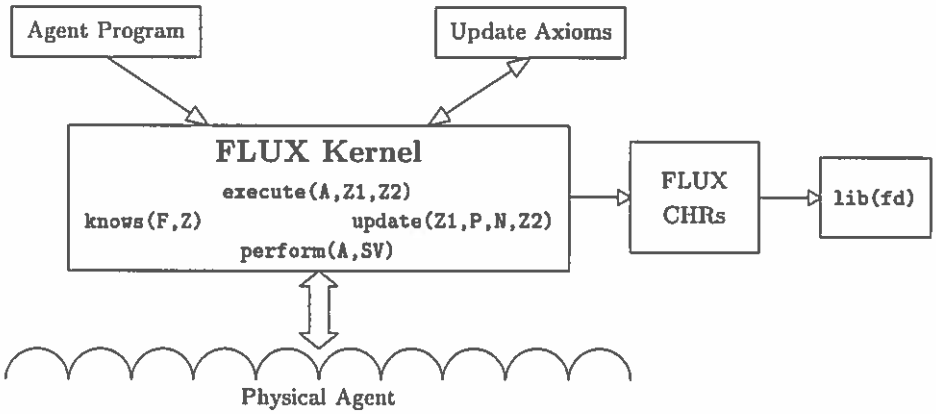


Figure 3: The architecture of a FLUX program. The actual performance of an action, triggered by *Perform*, connects the program to the effectors and sensors of the physical agent.

the specification of a search space (c.f. Section 2.4). To this end, the predicate $DO(g, s_0, s, z_0)$ defines situation s_0 plus action sequence s as a solution to planning problem g in state z_0. The definition follows closely the corresponding clauses in GOLOG [Levesque *et al.*, 1997]:

```
do(E,S0,S,Z0) :-
  E=[] -> S=S0 ;
  E=[E1|L] -> do(E1,S0,S1,Z0), do(L,S1,S,Z0) ;
  E=(E1#E2)-> (do(E1,S0,S,Z0); do(E2,S0,S,Z0)) ;
  plan_proc(E,E1) -> do(E1,S0,S,Z0) ;
  E=?(P) -> P =.. [Pred|Args],
            append(Args,[S0,Z0],ExtArgs),
            P1 =.. [Pred|ExtArgs], call(P1),
            S=S0 ;
  elementary_action(E) -> S=[E|S0] ;
  compound_action(E)   -> S=[E|S0].
```

Since at planning time the outcome of sensing actions cannot be predicted, all possibilities need to be taken into account. To this end, FLUX uses the predicate $Res(s, z_0, z)$, denoting that incomplete state z is a possible result of performing, in state z_0, the actions of situation s. In this way, a property of a future situation is known just in case there is no possible result in which the property could be false:

```
res([],Z,Z).
res([A|S],Z0,Z) :-
  res(S,Z0,Z1), state_update(Z1,A,Z,_).
knows(F,S,Z0) :-
  \+ ( res(S,Z0,Z), not_holds(F,Z) ).
knows_not(F,S,Z0) :-
  \+ ( res(S,Z0,Z), holds(F,Z) ).
```

Finally, given a specification of the cost of a plan, the search for an optimal plan is done by a standard search procedure that uses backtracking until no further solution can be found:

```
plan(Proc,Plan,Z0) :-
  assert(plan_search_best(void,0)),
  plan_search(Proc,Z0),
  plan_search_best(Plan,_),
  retract(plan_search_best(Plan,_)),
  Plan \= void.

plan_search(Proc,Z0) :-
  do(Proc,[],Plan,Z0),
  plan_cost(Proc,Plan,Cost),
  plan_search_best(BestPlan,BestCost),
  ( BestPlan \= void -> Cost < BestCost
                      ; true ),
  retract(plan_search_best(BestPlan,BestCost)),
  assert(plan_search_best(Plan,Cost)), fail
  ; true.
```

5 SOUNDNESS OF FLUX PROGRAMS

In this section, we give a brief overview of how properties of FLUX programs can be formally established with the help of the underlying Fluent Calculus semantics. We assume the reader to be familiar with basic notions and notations of logic programming with negation and constraints. In particular, we consider the standard left-to-right selection rule and so-called SLDNF-derivation trees, which consist of a main tree along with a number of subsidiary trees for negated literals [Apt and Bol, 1994].

The notion of a situation provides the semantics for the execution of a FLUX agent program.

Definition 2 Let T be the derivation tree for a FLUX program and query $P \cup \{Q\}$. An *execution node* in T is a node whose selected atom is $Init(\tau)$ or $Execute(\alpha, \tau_1, \tau_2)$. In the latter case,

term α is called the *executed action*. If N is a node in T, then the *situation associated* with N is $Do(\alpha_n, \ldots, Do(\alpha_1, S_0)\ldots)$ iff the path leading to N, but without N itself, satisfies the following:

1. the first execution node is of the form $Init(\tau)$;
2. no other execution node is of the form $Init(\tau)$; and
3. $\alpha_1, \ldots, \alpha_n$ is the ordered sequence of executed actions.

If the path does not contain an execution node, then the situation associated with N is S_0; in any other case the associated situation is undefined. □

The following notion of soundness describes a generally desirable property of FLUX agent programs. Informally speaking, all executed actions should be possible and there should be no backtracking over executed actions.

Definition 3 A FLUX program P with query Q is *sound* wrt. a Fluent Calculus axiomatization Σ iff each execution node $N = Execute(\alpha, \tau_1, \tau_2)$ in the computation tree for $P \cup \{Q\}$ satisfies the following:

1. α is a ground ACTION term;
2. the situation σ associated with N is defined and satisfies $\Sigma \models Poss(\alpha, \sigma)$;
3. N is in the main tree;
4. N lies on a branch that does not fail. □

Our agent program of Section 2 for the Wumpus World can be proved sound in this sense wrt. the Fluent Calculus axiomatization of Section 3.

Theorem 4 *The Wumpus World agent program with query $\{Main\}$ is sound.*

Proof (sketch):

1. The only occurrence of an execution node of the form *Init* is as the first atom in the body of the clause for *Main*.
2. All executed actions are constants of sort ACTION.
3. Action *Enter* is executed only as the first action, which is possible according to (2) and the precondition axioms.
4. Action *Exit* is executed only if the agent has either returned to square $(1,1)$ by unrolling the backtrack path or has planned and executed a route to this square. In both cases, the action is possible.
5. Action *Turn* is executed only after *Enter* and prior to *Exit*.
6. Action *Go* is executed only if there is an adjacent cell which is known to be free of a pit and if the Wumpus is known to be elsewhere or dead.
7. Action *Grab* is executed only if the agent is in a square that is known to contain gold.
8. Action *Shoot* is executed at most once, where the agent still has the arrow.
9. No negated literal depends on the atoms *Init* or *Execute* in the program; hence, all execution nodes are in the main tree.
10. The program always ends with success and there is no backtracking over execution nodes. ■

6 COMPUTATIONAL BEHAVIOR

Studies have shown that FLUX exhibits excellent computational behavior beyond problems of toy size. In the accompanying paper [Thielscher, 2002a], we report on experiments with a special variant of FLUX for complete states applied to a robot control program for a combinatorial mail delivery problem. The results show that FLUX can compute the effects of hundreds of actions per second. Most notably, the average time for selecting an action and inferring the effects remains essentially constant as the program progresses, which shows that FLUX scales up effortlessly to arbitrarily long sequences of actions. This result has been compared to GOLOG [Levesque *et al.*, 1997], where the curve for the computation cost suggests a polynomial increase over time. The analysis shows that the paradigm of a state-based representation is necessary for programs to scale up well to the control of agents and robots over extended periods of time: By maintaining an explicit state term throughout the execution of the program, fluents can be directly evaluated in FLUX programs, whereas an implicit state representation as in [Levesque *et al.*, 1997] or [Shanahan and Witkowski, 2000] leads to ever increasing computational effort as the program proceeds.

The computational behavior of FLUX in the presence of incomplete states has been analyzed in [Thielscher,

2002a] with a combinatorial problem that involves exploring a partially known environment and acting cautiously under incomplete information, much like in the Wumpus World. Although incomplete states pose a much harder problem, FLUX proves to scale up impressively well again: During a first phase, where the agent enhances its knowledge of the environment wandering around, there is a mere linear increase of the computation cost as the knowledge base grows. This result is particularly remarkable since the agent needs to constantly perform theorem proving tasks when conditioning its behavior on what it knows about the environment. Linear performance has been achieved due to a careful design of the state constraints supported in FLUX; the restricted expressiveness makes theorem proving computationally feasible. During the second phase of the aforementioned program, where the agent acts under the still incomplete knowledge, the average time for making decisions and inferring the effects of actions remains constant again. This shows that general FLUX, too, scales up effortlessly to long sequences of actions.

Planning with incomplete states, on the other hand, is a notoriously hard problem in FLUX, too. If the domain-dependent search space contains just a linear number of nondeterministic choices, there are exponentially many plans to be searched, and hence planning cannot scale up. Following an argument already put forward in [Giacomo and Levesque, 1999], the consequence for the agent programmer is that the planning facilities should be used restrictively and for bounded sub-problems only, in order not to spoil the computational merits of FLUX.

7 FUTURE WORK

Several crucial aspects of acting in real-world environments have not been tackled in this paper, such as the fact that actions may fail unexpectedly. An approach to this problem has been presented in [Thielscher, 2001b], where default assumptions regarding executability have been added to the theory of the Fluent Calculus. Incorporating this technique into agent programs is an important aspect of future work, as is modeling dynamic environments that constantly evolve around agents.

References

[Apt and Bol, 1994] K. Apt and R. Bol. Logic programming and negation: A survey. *Journal of Logic Programming*, 19/20:9–71, 1994.

[Bibel, 1986] W. Bibel. A deductive solution for plan generation. *New Gener. Comp.*, 4:115–132, 1986.

[Frühwirth, 1998] T. Frühwirth. Theory and practice of constraint handling rules. *Journal of Logic Programming*, 37(1–3):95–138, 1998.

[Giacomo and Levesque, 1999] G. De Giacomo and H. Levesque. An incremental interpreter for high-level programs with sensing. In H. Levesque and F. Pirri, editors, *Logical Foundations for Cognitive Agents*, pages 86–102. Springer, 1999.

[Hölldobler and Schneeberger, 1990] S. Hölldobler and J. Schneeberger. A new deductive approach to planning. *New Gener. Comp.*, 8:225–244, 1990.

[Levesque *et al.*, 1997] H. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1–3):59–83, 1997.

[Reiter, 2001] R. Reiter. *Logic in Action.* MIT Press, 2001.

[Russell and Norvig, 1995] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, 1995.

[Shanahan and Witkowski, 2000] M. Shanahan and M. Witkowski. High-level robot control through logic. In *Proceedings of ATAL*, volume 1986 of *LNCS*, pages 104–121, 2000. Springer.

[Thielscher, 1999] M. 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.

[Thielscher, 2000a] M. Thielscher. Modeling actions with ramifications in nondeterministic, concurrent, and continuous domains—and a case study. In *Proceedings of AAAI*, pages 497–502, 2000. MIT Press.

[Thielscher, 2000b] M. Thielscher. Representing the knowledge of a robot. In *Proceedings of KR*, pages 109–120, 2000. Morgan Kaufmann.

[Thielscher, 2001b] M. Thielscher. The qualification problem: A solution to the problem of anomalous models. *Artificial Intelligence*, 131(1–2):1–37, 2001.

[Thielscher, 2002a] M. Thielscher. Pushing the envelope: Programming reasoning agents. 2002. (Submitted).

[Thielscher, 2002b] M. Thielscher. Reasoning about actions with CHRs and finite domain constraints. 2002. (Submitted).

Applications and Implementation

A fuzzy set approach to flexible case-based querying: methodology and experimentation

Martine de Calmès[(*)], Didier Dubois[(*)], Eyke Hüllermeier[(**)], Henri Prade[(*)], Florence Sèdes[(*)]

[(*)] IRIT- Institut de Recherche en Informatique de Toulouse
Université Paul Sabatier, 118 route de Narbonne
F-31062 Toulouse cedex 4, France

[(**)] Computer Science Department
University of Dortmund
D-44221 Dortmund, Germany

Abstract

Queries to a database can be made more powerful, by allowing flexibility in the specification of what has to be retrieved, and by referring to cases either for expressing the request, or for computing the answer. In this paper, we present an implemented information system (applied to a database describing houses to be rent), based on an approach developed in the fuzzy set and possibility theory setting. This provides a unified framework for expressing users' preferences about what he is looking for, for weighting the importance of requirements, for referring to examples that he likes and/or counter-examples that he dislikes, and for making case-based predictions.

1 INTRODUCTION

There has been recently an increasing interest for recommender systems, e.g., (Resnick and Varian, 1997). There are different types of such systems. Many exploit observed correlations between individual customers. Others mainly take advantage of information on the preferences of the users. In both cases, fuzzy set-based approaches are of interest for modelling similarity relations between customers or objects, and for representing preference profiles, e.g., (Perny and Zucker, 2001; Yager, 2001).

In the following, we report on an information system which has been implemented in the framework of an experimental platform in information processing (http://www.irit.fr/PRETI) developed at IRIT in Toulouse. The database which is used stores information about 700 houses to be rent for vacations in the south of France. Each house is described in terms of about 25 attributes which may be binary, discrete or numerical.

Presently, three of the PRETI functionalities which make use of fuzzy set-based methods, deal with information querying:

i) flexible querying, where the user can express his preferences on the values of attributes of interest, and assess the level of importance of each elementary requirement when specifying what he is looking for;

ii) querying by examples, where the request is specified through the intensity of attraction (or repulsion) for some examples according to the user's wishes;

iii) case-based prediction, where an attribute value (e.g., the price) is fuzzily estimated from the specification of other attributes (e.g., the size and the location of the type of house the user is interested in), taking into account the cases stored in the base.

We now present these three functionalities, by each time giving the basics of the fuzzy set-based methodology which is used, and illustrating it on the application.

2 FLEXIBLE QUERIES

Introducing flexibility in database querying has become an important research issue (e.g., (Christiansen et al., 1997), (Larsen et al., 2001)). The main motivation for using flexible queries is the modelling of users' preferences. This has been advocated at length in the fuzzy database literature (e.g., Bosc and Kacprzyk, 1995; Petry, 1996).

Indeed fuzzy set membership functions (Zadeh, 1965) are a convenient tool for representing preference profiles, and moreover the large panoply of fuzzy set connectives can capture the different user attitudes concerning the way the levels of satisfaction of the different requirements present in the queries compensate or not; see (Bosc and Pivert, 1992) for a unified presentation in the fuzzy set framework of existing proposals for handling flexible queries.

2.1 REPRESENTATION ISSUES

Thus, the interest of fuzzy queries for a user is twofold:

i) A better representation of his preferences. For instance, "he is looking for an apartment which is *not too expensive* and *not too far* from downtown". In such a case, there does not exist a definite threshold for which the price becomes suddenly too high, but rather we have to differentiate between prices which are perfectly acceptable for the user, and other prices, somewhat higher, which are still more or less acceptable (especially if the apartment is close to downtown).

ii) Fuzzy queries, by expressing the user's preferences, provide the necessary information in order to rank-order the answers contained in the database according to the degree to which they satisfy the query. It contributes to avoid empty sets of answers when the queries are too restrictive, as well as large sets of answers without any ordering when queries are too permissive.

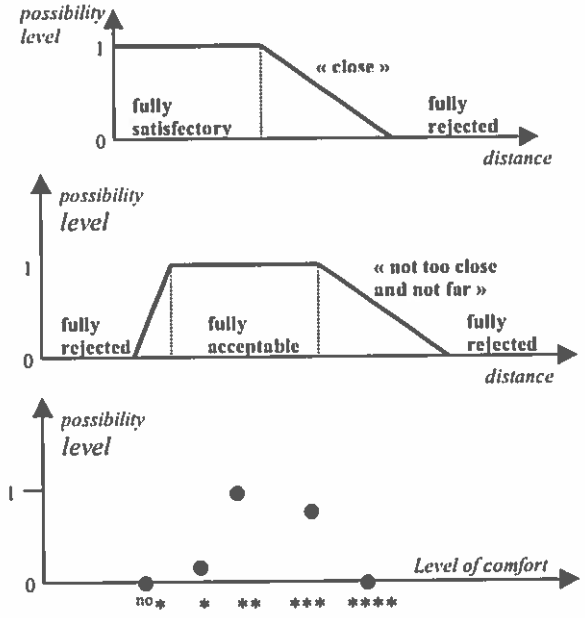


Figure 1. Examples of representations of fuzzy specifications

Obviously, the meaning of vague predicate expressions like "not too expensive" is context/user dependent, rather than universal. It reflects the own point of view of the user. So the user's intended meaning of such linguistic expressions has to be elicited. On continuously ordered attribute domains, it can be easily done in terms of areas of (non-ambiguous) values which are found fully acceptable, or which are fully rejected by the user, the other intermediate areas corresponding to gradual transitions of the possibility levels. On discrete domains, one should directly assess the possibility levels expressing how much satisfactory each value in the attribute domain is, according to the request. See Fig. 1.

Let P be a preference profile, and P(t) denote the possibility degree of having the considered attribute value equal to t. P(t) estimates to what extent it is acceptable to have the considered attribute equal to t. Let 1 and 0 denote the top and bottom elements of the possibility scale. P(t)=1 means that t is fully acceptable while P(t)=0 means that t is fully rejected. The larger P(t), the more acceptable t. Here we use the scale [0,1] for grading the possibility levels.

2.2 IMPORTANCE WEIGHTING

In a given query, some part of the request may be less important to fulfil than others; this leads to the need for weighted connectives. Thus, an assessment of the importance of the (soft) constraints involved in a request can be taken into account. Especially, using a min-based conjunction, several types of importance weighting, having different semantics can be used; see, e.g., (Dubois, Nakata and Prade, 1999).

Thus we may distinguish between three basic ways of modifying an elementary fuzzy request (pertaining to one attribute) by a weight. The weight can be seen as i) a discounting factor, ii) a thresholding factor, or iii) a bonus factor. Let $w \in [0,1]$ be a weight. A profile P can be modified by w using the following expressions according to the semantic nature of w:

- max(1−w, P(t))	(discounting)
- 1 if P(t) ≥ w, P(t) if P(t) < w	(thresholding)
- min (1, 1−w + P(t))	(bonus)

As it can be seen, if w=1, the expression yields P(t) in each case: if the attribute has the maximum level of importance, there is no effect on the evaluation provided by P(t). If w=0, the above expressions yield 1, which means that if the attribute has a zero importance level, the evaluation P(t) would no affect the conjunctive aggregation process of the elementary evaluations (since 1 is a neutral element for conjunctive operators, i.e. operators such as min, or smaller than min, like as the product).

The semantics of weight w depends on the expression used. In the discounting case, the evaluation cannot go below $1-w$ even if P(t)=0; it means that the violation of a not too important requirement cannot lead to rejection, since it cannot cause the acceptability level to go below $1-w$ (the more important the requirement, the smaller $1-w$). Thresholding means than it is only

necessary to reach level w to be considered as fully satisfactory. Bonus means that the less important the requirement, the greater the bonus (equal to $1-w$) which is added to the evaluation. Note that the two first operations are qualitative in nature since they can be defined as well on a discrete linearly ordered possibility scale, while the last one requires a numerical scale (at least a (bounded) integer-like scale). In this paper, we shall view the lack of importance as a discounting factor only.

Let $P_1,\ldots, P_n$ be the elementary requirements contained in a request, and let $P_i(a_i(x))\in[0,1]$ be the satisfaction degree of P_i by the item x where $a_i(x)$ is the value of attribute i for the item x. Then the conjunctive combination of these degrees, is defined as

$$c(t) = \min_i P_i(t).$$

Here we use the min operation for the conjunction because this is the greatest associative non-decreasing operation $*$ such as $a*1=a$ and $0*0=0$. It is also the only idempotent one. It means that the smallest elementary evaluation sanctions the global one.

This combination can be weighted in the following way

$$c(x) = \min_i \max(1 - w_i, P_i(a_i(x)))$$

where $w_i\in[0,1]$ is the level of importance of requirement P_i. Moreover it is assumed that $\max_i w_i=1$, i.e. the most prioritary requirements have a maximal level of importance and their violation leads to a zero score. Again, when $w_i=1$, the importance of attribute i is maximal. For $0 < w_i < 1$, a bad score for $P_i(t)$, i.e. $P_i(t)$ close to 0 cannot cause the global evaluation to go below $1-w_i$, so the effect of a bad score for $P_i(t)$ is limited and controlled.

2.3 RELATION TO DECISION FRAMEWORKS

The expression of c(t) is technically speaking the necessity measure of a fuzzy event (Dubois and Prade, 1980). Indeed, let us consider the set functions N from $2^{\{1,\ldots,n\}}$ to [0,1], such as $N(A\cap B)=\min(N(A), N(B))$ where A and B are subsets of $\{1,\ldots,n\}$.

In our context, N is defined from the possibility distribution π given by $\pi(i)=w_i$; its expression is

$$N(A)=\min_{i\notin A}(1-\pi(i))=\min_{i\notin A}(1-w_i).$$

N(A) estimates to what extent A contains important attributes only: $N(A)=0$ as soon as $\exists$ i, $w_i=1$ and $i\notin A$. N is associated with a possibility measure Π by the duality equation $N(A)=1-\Pi(\bar{A})$, where $\Pi(A)$ estimates to what extent A contains at least one important attribute. N can be extended to a fuzzy set F defined as a fuzzy set of $\{1,\ldots,n\}$ by the expression

$$N(F)=\min_{i=1,\ldots,n}\max(F(i), 1-\pi(i)).$$

It estimates the extent to which all the important attributes are among the i such as F(i) is high, i.e. in c(x), the extent to which all the important attributes get a high score for the considered item t.

The expression of c(x) is also close to a qualitative expected utility. Indeed, viewing the w_i's rather as degrees of uncertainty, let us imagine for instance that the w_i's no longer assess the importance of the attribute i, but assess the possibility of being in a state i. Then the possibility distribution π (defined by $\pi(i)=w_i$) represents the set of more or less possible states which can be reached for some decision according to the available information. Viewing F(i) as the utility of being in state i, N(F) is high for decision such that all their possible consequences have a high utility value. Von-Neumann-like and Savage-like justifications of this decision index can be found in (Dubois, Godo et al., 1999) and (Dubois, Prade and Sabbadin, 2001) respectively. See also (Dubois, Fargier et al., 2001) and (Dubois, Marichal et al., 2001) for a parallel between decision under uncertainty and multiple-criteria decision, and a discussion of c(t) in a multiple-criteria decision perspective.

Besides, c(x) can be extended to the case where the database contains imprecisely/fuzzily-known attribute values also represented by possibility distributions. Then c(x) is replaced by two evaluations, the extent to which item x satisfies *necessarily* all the important requirements, and the extent to which item satisfies *possibly* all the necessary requirements according to the available information on the attribute values. This enables us to accommodate missing values of applicable attributes. Then the incompletely known items (w.r.t. to the considered attributes) can only *possibly* satisfy the request. See (Dubois, Prade and Testemale, 1988) for details and justifications of this type of weighting.

2.4 AGGREGATION FUNCTIONS

We may think of scoring rules other than c(x). The simplest one would be a weighted average of the form $\sum_{i=1,n} w_i\cdot P_i(a_i(x))$. Such a scoring rule embeds compensatory effects between bad and good scores, which may be desirable or not. However, compensation is not qualitative in nature. Finding out if the user allows or not for some compensatory effects between the satisfaction levels of the different parts of his request, would require efforts to be elicitated. It appears simpler to use, by default, a more drastic aggregation process based on min operation which calls for the satisfaction of each elementary requirement. We may think that, in some cases, the min-based aggregation leads to a ranking of retrieved items that is insufficiently discriminating because it relies on the comparison of the least satisfied properties.

Flexible Query Submit

Importance ([0,1])	Attribute	Value
1	ACCOM. NB	*At least* 6
0.3	DISHWASHER	yes
1	BEACH	Distance (km) *acceptable from* 0 , *preferably between* 0 *et* 1 , *in any case not farther than* 3
1	SHOPS	Distance (km) *acceptable from* 0 , *preferably between* 0 *et* 3 , *in any case not farther than* 5

Figure 2. Example of a flexible query with 4 attributes

Indeed min-based aggregation is often criticized for its lack of sensitivity to the change of the arguments. However refinements of the min operator have been proposed, namely the discrimin and the leximin orderings (Dubois et al., 1997). Let $(u_1,\ldots, u_n)$ and $(v_1,\ldots, v_n)$ be two vectors of evaluations. Then

$$(u_1,\ldots, u_n) <_{\text{discrimin}} (v_1,\ldots, v_n)$$

$$\text{iff } \min_{i\in D(u,v)} u_i < \min_{i\in D(u,v)} v_i$$

where $D(u,v) = \{i | u_i \text{ not equal to } v_i\}$ is the discriminating set. This clearly refines the min-based order. The leximin which refines the discrimin is based on the lexicographic comparison of the vectors once their components are increasingly reordered. The PRETI system implements these two refinements for improving the ordering of the retrieved items in the list provided to the user.

Recently, Fagin and Wimmers (2000) have advocated the use of another weighted aggregation mode, namely,

$$\sum_i (w_i - w_{i+1}) \cdot i \cdot f(x_1,\ldots,x_i)$$

where $\sum_i w_i=1$ and $w_1 \geq \ldots \geq w_i \geq \ldots \geq w_n$ and f is an aggregation function. The advantage of this scheme is its generality; moreover, it enjoys a restricted linearity property with regard to the weights w_i which makes it similar to a Choquet integral (see, e.g. (Grabisch et al, 1995)). However, this framework requires a numerical scaling, while the weighted min aggregation only requires an ordinal, linearly ordered scale ($1-(\cdot)$ being the order-reversing map of the scale).

Sugeno integral (e.g., Dubois, Marichal et al., 2001) can be seen has a qualitative counterpart of Choquet integral, however, what makes them attractive is the possibility to weight the requirements not only individually but also in subgroups. This enables to take into account some synergy between the requirements. However it appears here far too sophisticated w.r.t. to users' needs.

Besides, the weighted min evaluation might be also improved by requiring only the satisfaction of *most* requirements. This can be done by using an *ordered* weighted min expression where the higher weights go to the higher evaluations $P_i(a_i(x))$; see, e.g., (Dubois *et al.*, 1988) for details.

2.5 EXPERIMENTATION AND RANKING

Fig. 2 shows a request expressing a search for an house with at least 6 beds, preferably at less than 1 km from the sea and in any case not farther than 3 km; a condition of the same type is expressed about the distance to shops. All the requirements have the same importance except the last one about the wish for a dish-washing machine.

10 houses match your request
3 perfectly

Matching Degree	House Identification	Place	Request Attribute Values
1.000	571-11	GRUISSAN	Accommodation Number: 6 Dishwasher: yes Beach Dist.: 0 km 500 Shop Dist.: 1 km
1.000	538-11	FLEURY-D'AUDE	Accommodation Number: 6 Dishwasher: yes Beach Dist.: 0 km 300 Shop Dist.: 0 km
1.000	156-11	LEUCATE	Accommodation Number: 6 Dishwasher: yes Beach Dist.: 0 km 500 Shop Dist.: 1 km 500
0.750	182-11	LA PALME	Accommodation Number: 6 Dishwasher: yes Beach Dist.: 1 km 500 Shop Dist.: 1 km 500
0.700	468-11	GRUISSAN	Accommodation Number: 6 Dishwasher: no Beach Dist.: 1 km Shop Dist.: 0 km 500
0.700	151-11	LEUCATE	Accommodation Number: 6 Dishwasher: no Beach Dist.: 1 km 500 Shop Dist.: 0 km 500
0.500	455-11	LEUCATE	Accommodation Number: 6 Dishwasher: yes Beach Dist.: 2 km Shop Dist.: 1 km

Figure 3. Answers of the first query

For simplicity the levels of importance are numerically assessed on the screen, however, a linguistically interpreted scale would be used for interface with an end-user.

Fig. 3 is an excerpt of the list of the 10 houses retrieved by the system ; 3 perfectly match the query. Note that because the dish-washer is not so important, houses without one but well located w.r.t. sea are preferred to houses which are farther, even with a dish-washer.

By making an aggregation (here based on min operation) of the degrees of acceptability according to the different requirements on the scale [0,1], we make a commensurateness hypothesis between the different scales for assessing the degrees of acceptability w.r.t. to the different elementary requirements. Moreover the scale used for importance levels is also mapped on the scale [0,1]. The merit of the commensurateness hypothesis is clearly its simplicity, since it provides a complete ordering of the answers to a request. This ordering thus stratifies the set of answers into layers corresponding to different satisfactory levels. It might be enough, and even clearer for the user, to use a scale with a finite (not too large) number of levels. In case of preferences profiles defined on continuous domains, it just means that the degrees $P_i(a_i(x)) \in [0,1]$ would have to be approximated to the closest level in the finite scale (supposed to be encoded by a finite number of values regularly distributed between 0 and 1). Thus, the user is faced with a smaller number of categories of answers having different levels of acceptability, assuming a conjunctive, non-compensatory aggregation.

5 houses match your request
1 perfectly

Matching Degree	House Identification	Place	Request Attribute Values
1.000	156-11	LEUCATE	Accommodation Number: 6 Comfort Level: 4 stars Dishwasher: yes Beach Dist.: 0 km 500 Shop Dist.: 1 km 500
0.750	182-11	LA PALME	Accommodation Number: 6 Comfort Level: 4 stars Dishwasher: yes Beach Dist.: 1 km 500 Shop Dist.: 1 km 500
0.500	571-11	GRUISSAN	Accommodation Number: 6 Comfort Level: 3 stars Dishwasher: yes Beach Dist.: 0 km 500 Shop Dist.: 1 km
0.500	538-11	FLEURY-D'AUDE	Accommodation Number: 6 Comfort Level: 3 stars Dishwasher: yes Beach Dist.: 0 km 300 Shop Dist.: 0 km
0.500	455-11	LEUCATE	Accommodation Number: 6 Comfort Level: 3 stars Dishwasher: yes Beach Dist.: 2 km Shop Dist.: 1 km

Figure 4. Answers with the refined query

Then additional requirements can be used for refining a given set of answers. Going back to our example, let us assume that the user is moreover looking for a top quality house. Assume that he finds **** houses perfectly acceptable and *** houses half acceptable. Then there remains only one perfectly acceptable house, while the two other perfect matches of Fig. 3 are now discounted, and other houses which are only ** or less have disappeared. See Fig. 4. A theoretical discussion of qualitative decision based on a hierarchy of subsets of criteria (as in this example with the additional criterion) can be found in (Dubois, Fargier, Perny, 2002).

2.6 AVOIDING EMPTY SETS OF ANSWERS

One of the main advantages of allowing for flexible queries is to diminish the risk of empty sets of answers by enlarging the request from the beginning by taking into account the tolerance of the user for going away from what is fully satisfactory for him. However, it may still happen that all the items x in the base are such that $\min_{i \in I} P_i(a_i(x))=0$ where I is the set of attributes having the maximum level of importance. It is possible to avoid empty sets of answers to a large extent by providing the user with the answer that there exists no house (in our example) which somewhat fits his requirements (which may be contradictory in practice, e.g. being close both to the sea and to the mountain), but indicating him the closest answers.

Let us sketch how it can be done. The idea is to equip each elementary requirement pertaining to an ordered domain with a default preference profile P_i' such that P_i' coincides with P_i on the values where $P_i(t)=1$, and is non zero elsewhere on the whole attribute domain, with the larger $P_i'(t')$, the closer t' is to the set {t such that $P_i(t)=1$}. Thus when browsing the base, we can incrementally keep the closest answer for all the situations where one of the most important attributes is not at all satisfied, and computing the degree of acceptability restricted to the remaining attributes. This approach could be extended to the case where more than one fully important attribute is violated.

3 QUERYING BY EXAMPLES

The user is not always able to easily express his request by expressing restrictions on attribute domains, even in a flexible way, these restrictions being then conjunctively combined. It may be more convenient for him to express what he is looking for from examples or prototypes (that are proposed to him by the system, or which are generated by himself). Querying based on examples, for eliciting the user's preferences, can provide the necessary information for building a query. Thus, the user may say to what extent a few examples of items are representative of what he is looking for by

using some (finite) preference scale. Then, the relevance of stored items should be evaluated in terms of their similarity with respect to these preferred examples (and may be dissimilar from counter-examples he dislikes). Issues are then close to fuzzy case-based reasoning (Dubois et al., 1998).

Examples, as well as counter-examples, are supposed to be described in terms of precisely known attribute values. These attributes are assumed to be relevant and sufficient for describing their main features from a user's point of view. Then a request should look for the items which are similar to at least one example (w.r.t. all the attributes) and which are dissimilar to all the counter-examples (each time w.r.t. at least one attribute), as proposed in (Dubois, Prade, Sèdes, 2001).

Moreover we can take into account the extent to which each example or counter-example is highly representative, and the importance of the (relevant) attributes. Thus, the more *similar* an item x is w.r.t. *(at least)* a representative example and the more *dissimilar* x is w.r.t. *all* representative counter-examples, the more *possible* x is eligible for the user. This evaluation might be also improved by requiring the similarity of x with *most* examples only (see (Dubois *et al.*, 1988) for the handling of softly quantified weighted min expression).

Suppose for simplicity that all the importance and representativeness weights are equal, the items x are then rank-ordered in terms of the function

$$c(x) = \min[\max_j \min_i S_i(a_i(x), a_{ij}),$$
$$\min_k \max_i 1 - S_i(a_i(x), b_{ik})],$$

where $a_i(t)$ is the value of attribute i for item x, a_{ij} is the value of attribute i for example j, b_{ik} the value of attribute i for counter-example k, where S_i is a fuzzy similarity relation on the attribute domain of i (S_i is supposed to be reflexive and symmetric). Clearly, $S_i(\cdot, a_{ij})$ defines a fuzzy set of values close to a_{ij}, while $1 - S_i(\cdot, b_{ik})$ defines a fuzzy set of values not similar to b_{ik}, for each attribute i (it may be replaced by a smaller fuzzy set of values *significantly* different from b_{ik}).

Note also that the above expression may incorporate *interactivity* between attributes. For instance, the user may be simultaneously interested in two types of houses, one which is 'small' with a 'low' price, and another which is 'large' but with a 'medium' price. Besides, we may think of hybrid queries made of examples and of classical (fuzzy) restrictions expressing what attribute values are undesirable.

Fig. 5 exhibits the description of 5 typical houses according to 4 attributes (occupancy, distance to the beach, distance for sailing, price), chosen as relevant by the user. Moreover he can state the importance of these criteria linguistically. So the user assesses how representative of what he likes (or dislikes) is the house, i.e. he provides the intensity (on a 6-levels scale) with which he likes, or he dislikes the house. He may also choose to be indifferent. In this example, he likes house number 1 at the maximum level, house number 2 at a smaller level. He hates house number 4 and he is indifferent to the others.

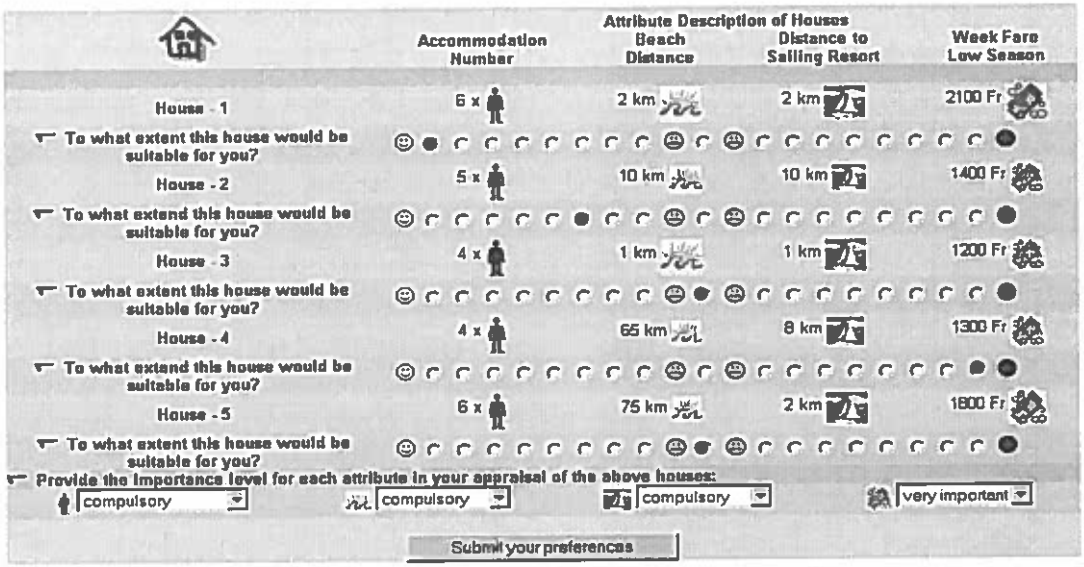


Figure 5. Assessment of preferences on 5 potential examples

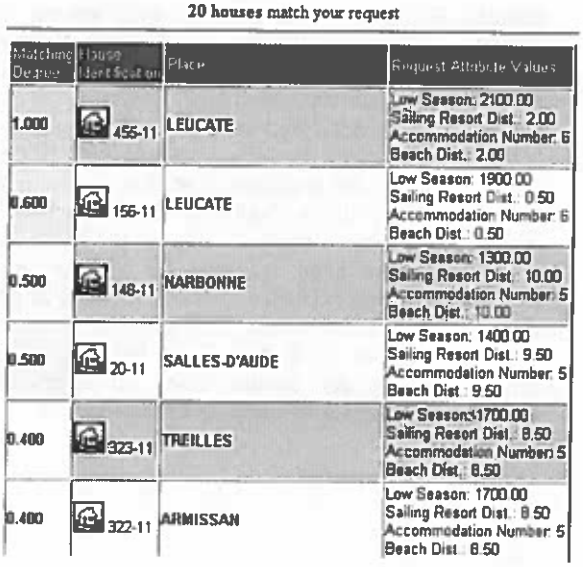


Figure 6 Answers for the case-based querying

Fig. 6 shows the 6 best answers to the queries induced by the preferences expressed in Fig. 5, among 20 retrieved houses. The system uses similarity measures which are assessed by default for each attribute domain; they may be however tuned by the user. We can observe that in practice the filtering is mainly based on the liked examples. As it can be guessed, in general the counter-examples (the houses which are disliked) have only an impact if no positive example is given, or if there is some inconsistency in the user's preferences (this may happen if examples which are too close are given with opposite opinions about them).

Note that the houses which are used as potential examples could be:

- either obtained as prototypical houses in the base which are sufficiently different from each other with respect to the considered attributes;
- or described by the user. Once he has selected the important attributes, he may provide fictitious examples of houses he would like to find.

In some cases, it may happen that the positive and negative preferences expressed by the user are somewhat contradictory. For instance assume that we have only two identical cases in the set of potential examples, and that the user says, by mistake, that he likes the first one and dislikes the second one very much, as in Fig. 7. In such a case the c(x) is of the form min(y, 1–y) and remains below 0.5 because of the contradiction. In our example we retrieve only one house in Fig. 8 with a score 0.1 (because this house would get the score 0.9 in case the user only indicates that he likes the house).

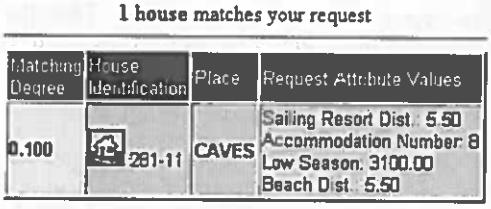


Figure 8 Answer with a contradictory assessment

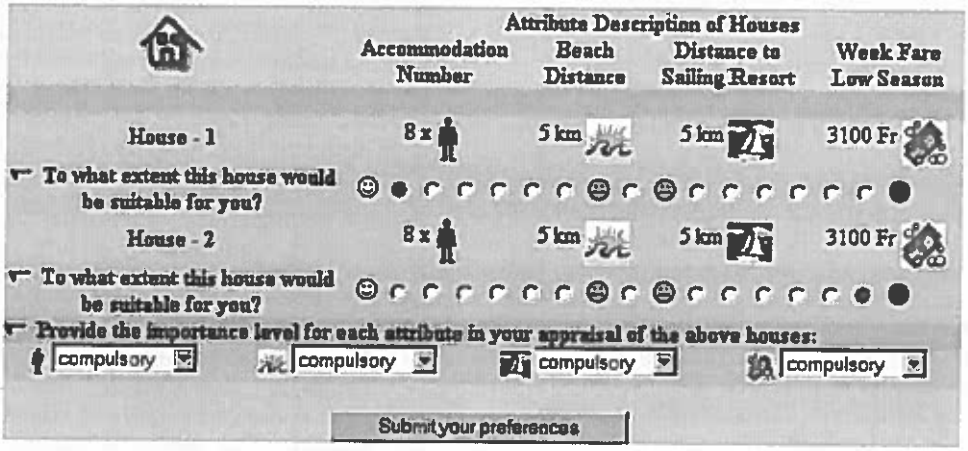


Figure 7. Example of a contradictory assessment

4 CASE-BASED PREDICTION

Case-based reasoning, in general, assumes the following implicit principle: "*similar situations may give similar outcomes*". Thus, a similarity relation S between problem descriptions or situations, and a similarity measure T between outcomes are needed. This implicit CBR-principle can be expressed in the framework of fuzzy rules as, "*the more similar are the values of the situation attributes in the sense of S, the more possible the similarity of the values of the outcome attributes in the sense of T*" (Dubois *et al.*, 1998). Given a situation s_0 associated to an unknown outcome t_0 and a current case (s,t), this principle enables us to conclude on the *possibility* of t_0 being equal to a value similar to t. This acknowledges the fact that, often in practice, a database may contain cases which are rather similar with respect to the problem description attributes, but which may be distinct with respect to outcome attribute(s). This emphasizes that case-based reasoning can only lead to cautious conclusions.

This can be modelled in terms of the possibility rule (Dubois and Prade, 1996) "the more similar s and s_0, the more *possible* t and t_0 are similar". Then the fuzzy set F of possible values t' for t_0 with respect to case (s, t) is given by

$$F_{t_0}(t') = \min(S(s, s_0), T(t, t')).$$

As it can be seen, what is obtained is the fuzzy set $T(t, \cdot)$ of values t' which are T-similar to t, whose possibility level is upper bounded by the global degree $S(s,s_0)$ of similarity of s and s_0. The max-based aggregation of the various contributions obtained from the comparison with each case (s,t) in the memory M of cases, acknowledges the fact that each new comparison may suggest new possible values for t_0. Thus, we obtain the following fuzzy set E_{s_0} of possible values t' for t_0 :

$$E_{s_0}(t') = \max_{(s,t)\in M} \min(S(s,s_0), T(t,t')).$$

This latter expression can be put in parallel with the evaluation of a flexible query. Indeed here a situation is described in terms of a collection of attributes, and we are looking for cases which are constrained to be similar to s_0 w.r.t. the attributes. But rather than exhibiting the fuzzy set of cases similar to s_0, we produce the fuzzy set of the fuzzy sets of values similar to the t-attribute values of these cases (a level 2 fuzzy set reduced there to an ordinary fuzzy set by performing the union of the fuzzy sets of similar values). See (Dubois, Hüllermeier and Prade, 2000) for more details, for a generalization of the approach to situations s_0 imprecisely or fuzzily described, and for a comparative discussion with other approaches to instance-based prediction.

Besides, E_{s_0} may be a function with several local maxima. For instance, the situations s correspond to the description of products, and t to their prices; it may happen that products which are similar w.r.t. some subset of attributes (used in the specification of s_0) have different prices. Providing the user with E_{s_0} calls immediately for the explanation of this matter of fact. Assume that the values of other attributes which are not involved in s or t are also available for the cases stored in the memory. Then we may try to find out an explanation of the existence of several classes of prices in terms of those attributes. This amounts to compute the fuzzy set of attributes which take significantly different values for similar cases (in terms of s-attributes) having significantly different t-values.

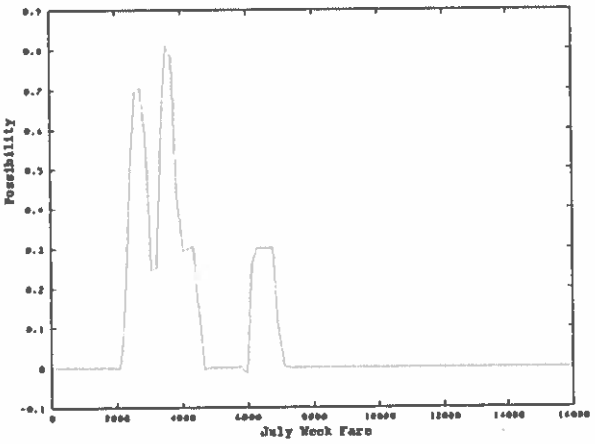


Figure 9. Possibility distribution of prices

Fig. 9 exhibits a possibility distribution of prices (in FF) when the user is inquiring for the possible prices of an house with at least 6 beds, at approximately 5 km from the station and 2 km from the sea. Further investigations of the user would enable him to discover that the leftest peak corresponds to houses without dish-washing machine while the rightest one corresponds to houses which can accommodate larger groups of people and have a dish-washer. In fact, this could be found out in an automatic way, using an approach sketched in (de Calmès et al., 2002).

Besides, we might think of replacing the max operation in the above expression by a sum (divided by the number of retrieved cases) in order to reflect the number of similar cases having some value t' (or a value close to t'). However, no difference will be made between a sum of low degrees (reflecting several not very similar cases) and a case, maybe unique, with a much higher degree of similarity (equal to this sum of low degrees). The above cardinality issue may be handled in more informative way, using fuzzy cardinality. This can be related to the following issue.

The approach presented in this section could be generalized in order to provide flexible case-based recommendation. For example suppose that the owner of an house wants to propose it at an attractive price, he may be interested in finding a price "which is slightly lower than the prices for most of the houses in the area which have similar comfort". The above approach can provide a view of possible prices of houses having some location and a given level of comfort (even if fuzzily specified). The supplementary evaluation which is required by the above query is to have some fuzzy statistics about the number of houses having some range of prices corresponding to a peak of the distribution like the one of Fig. 9. In the above notation, the problem is to evaluate the fuzzy-valued cardinality of the fuzzy set of cases similar to s_0 and whose t-values are close to the values in the considered peak (here the one of smaller prices). See (Bosc et al., 2001) for a preliminary study of how such a fuzzy cardinality can be computed.

5 CONCLUSION

When querying a database, the user may have intricated goals in mind: retrieving items corresponding to some requirements, referring to similar cases, using stored data for prediction purposes, trying to figure out range of attribute values for classes of items, looking for explanations. In all these operations flexibility can be introduced with profit, as shown by the implemented system.

The fuzzy logic-based techniques demonstrated in this paper could be still more interesting for querying multimedia databases, as advocated in (Fagin, 1998) or in (Dubois, Prade, Sèdes, 2001).

Besides, it may be also interesting to equip a system such as PRETI with some visualisation facilities. For instance, we may think of visualizing the locations of a set of houses which more less satisfy some flexible requirements on a map. In order to figure out the degrees of matching, we may use dots of variable sizes (the higher the matching degree, the larger the dot, here using an idea initially suggested by (Lévy, 1988)).

References

P. Bosc, J. Kacprzyk (Eds.) (1995) *Fuzziness in Database Management Systems.* Physica-Verlag, Heidelberg.

P. Bosc, O. Pivert (1992) Some approaches for relational databases flexible querying. *J. of Intelligent Information Systems*, 1, 323-354.

P. Bosc, O. Pivert, D. Dubois, H. Prade (2001) On Fuzzy Association Rules Based on Fuzzy Cardinalities. *Proc. of the 10th IEEE International Conference on Fuzzy Systems.*

M. de Calmès, D. Dubois, E. Hüllermeier, H. Prade, F. Sèdes (2002) Case-based querying and prediction: a fuzzy set approach, *Proc. of 11th IEEE Conf. on Fuzzy Systems*, Honolulu.

H. Christiansen, H.L. Larsen, T. Andreasen (Eds.) (1997) *Flexible Query Answering Systems*, Kluwer Academic Publishers.

D. Dubois., F. Esteva, P. Garcia, L. Godo, R. Lopez de Mantaras, H. Prade (1998) Fuzzy set modelling in case-based reasoning. *Int. J. of Intelligent Systems*, 13, 301-374.

D. Dubois, H. Fargier, P. Perny (2002) On the limitations of ordinal approaches to decision-making, *Eighth International Conference on Principles of Knowledge Representation and Reasoning* (KR2002), April 22-25, Toulouse, France.

D. Dubois, H. Fargier, P. Perny, H. Prade (2001) Towards a qualitative multicriteria decision theory. In *Proceedings of Eurofuse Workshop on Preference Modelling and Applications*, Granada, Spain, 121-129.

D. Dubois, H. Fargier, H. Prade (1997) Beyond min aggregation in multicriteria decision : (ordered) weighted min, discri-min, leximin. In *The Ordered Weighted Averaging Operators. Theory and Applications*, (R.R. Yager, J. Kacprzyk, eds.), Kluwer Acad. Publ., 1997, 181-192.

D. Dubois, L. Godo, H. Prade, A. Zapico (1999) On the possibilistic decision model: from decision under uncertainty to case-based decision. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 7, 631-670.

D. Dubois, E. Hüllermeier, H. Prade (2000) Flexible control of case-based prediction in the framework of possibility theory. In: *Advances in Case-Based Reasoning* (Proc. 5th Eureopean Worshop, EWCBR 2000, 6-9 sept.), Trento, Italie. (E. Banzieri, L. Portinal, Eds., LNCS n°1898), Springer-Verlag, Berlin Heidelberg , 61-73.

D. Dubois, J.L. Marichal, H. Prade, M. Roubens, R. Sabbadin (2001) The use of the discrete Sugeno integral in decision-making: a survey. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9, 539-561.

D. Dubois, M. Nakata, H. Prade (1999) Extended Divisions for flexible queries in relational databases. In: *Knowledge Management in Fuzzy*

Databases, (O. Pons, M. A. Vila, and J. Kacprzyk, Eds.), Physica-Verlag, Heidelberg, 105-121.

D. Dubois, H. Prade (1980) *Fuzzy Sets and Systems, Theory and Applications*, Academic Press.

D. Dubois, H. Prade (1996) What are fuzzy rules and how to use them. *Fuzzy Set and Systems*, 84, 169-185.

D. Dubois, H. Prade, R. Sabbadin (2001) Decision-theoretic foundations of qualitative possibility theory. *European Journal of Operational Research*, Elsevier, 128, 459-478.

D. Dubois, H. Prade, F. Sèdes (2001) Fuzzy logic techniques in multimedia database queyring: A preliminary investigation of the potentials. *IEEE Transactions on Knowledge and Data Engineering*, 13(3), 383-392.

D. Dubois, H. Prade, C. Testemale (1988) Weighted fuzzy pattern matching, *Fuzzy Sets and Systems*, 28, 313-331.

R. Fagin (1998) Fuzzy queries in multimedia database systems. *Proc. 7th ACM Symp. on Principles of Database Systems*, Seattle, 1-10.

R. Fagin, E.L. Wimmers (2000) A formula for incorporating weights in scoring rules. *Theoretical Computer Science*, 239, 309-338.

M. Grabisch, H.T. Nguyen, E.A. Walker (1995) *Fundamentals of Uncertainty Calculi with Applications to Fuzzy Inference*. Kluwer Acad.

J. Kacprzyk., A. Ziolkowski (1986) Data base queries with fuzzy linguistic quantifiers. *IEEE Trans. on Systems, Man and Cybernetics*, 16(3), 474-478.

H. Larsen, J. Kacpryk, S. Zadrozny, T. Andreasen, H. Christiansen (Eds.) (2001) Flexible Query Answering Systems, Recent Advances, Physica Verlag.

P. P. Lévy (1988) Graduated substrata, *Information Processing & Management*, 24, 693-702.

P. Perny, J.D. Zucker (2001) Preference-based search and machine learning for collaborative filtering: the "Film-Conseil" movie recommender system. *Information, Interactio, Intelligence*, 1 (1), 9-48.

F.E. Petry (1996) *Fuzzy Databases: Principles and Applications*. Kluwer Acad. Pub.

P. Resnick, H.R. Varian Recommender systems. *Communications of the ACM*, 40 (3), 1997.

R.R. Yager (2001) Fuzzy logic methods in recommender systems. Tech. Rep. MII-2116, Iona College, New Rochelle, NY.

L.A. Zadeh (1965) Fuzzy sets. *Information and Control*, 8, 338-353.

L.A. Zadeh (1978) Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1, 3-28.

A Generic Approach for Knowledge-Based Information-Site Selection

Thomas Eiter, Michael Fink, Giuliana Sabbatini, and Hans Tompits

Institut für Informationssysteme, Technische Universität Wien,
Favoritenstraße 9-11, A-1040 Vienna, Austria
e-mail: {eiter,michael,giuliana,tompits}@kr.tuwien.ac.at

Abstract

With the advent of the World Wide Web, a vast number of heterogenous information sources has become available. In order to access and process these data, suitable tools and methods for building an information infrastructure are necessary. One task for this purpose is the selection of relevant information sources in automated query answering. In this paper, we present an approach for information-site selection based on the answer set programming paradigm. We use extended logic programs to represent rich descriptions of the information sources, an underlying domain theory, and queries in a formal language. Moreover, we use extended logic programs also for declarative query analysis and query abstraction. The central part of our architecture are *site-selection programs*, representing both qualitative and quantitative criteria. Because of the structured nature of data items, the semantics of such programs, given in terms of prioritized logic programs, must carefully respect implicit context information in site-selection rules, and furthermore combine it with possible user preferences. An experimental prototype of the overall approach has been implemented using the `dlv` KR system and its `plp` front-end for prioritized logic programs.

1 Introduction

With the advent of the World Wide Web, a wealth of information has become available to a large group of users. However, the distribution of the information sources, their heterogeneity and diverse ways of accessibility have raised the need for suitable tools and methods for building an information infrastructure. A promising approach is based on multi-agent systems, in which special *information agents* (or *middle agents* [10]) provide various services, including finding, selecting, and querying relevant information sources.

The potential of knowledge-based approaches—especially of logic programming—for developing reasoning components of intelligent information agents is outlined in [14, 16, 29]. In this paper, we pursue this further and present a declarative approach to information source selection. Here, the problem is, given a query of the user, which out of a collection of information sources should the agent select for answering the query, such that the utility of the answer, in terms of quality of the result and other criteria (e.g., costs), is as large as possible for the user? Obviously, a sensible solution to this problem is nontrivial and involves various aspects such as elementary properties of the information sources, knowledge about their contents, and knowledge about the application domain.

Our approach incorporates such aspects, and makes several contributions, which are briefly highlighted as follows.

(1) We use extended logic programs [21] to represent rich descriptions of the information sources, an underlying domain theory, and queries in a formal language. We perform query analysis by logic programs and compute *query abstractions*. Here, we consider XML-QL [13], but other query languages could be handled as well.

(2) At the heart, a declarative *site selection program* represents *both qualitative and quantitative* criteria (such as site preference and costs). Its rules may access information provided by the other programs, including occurrences of objects and values in the query. For instance, a rule r_1 may state that a query about actors's awards (where actors

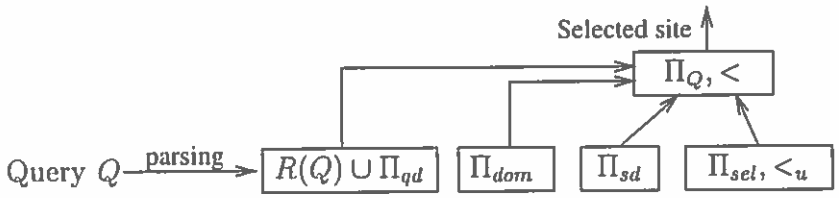


Figure 1: General Architecture of a Selection Base $\mathcal{S} = \langle \Pi_{qd}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u \rangle$ for Site Selection.

and awards are objects) should be posed to information source s_1.

(3) We consider the interesting and, to our knowledge, novel issue of *contexts* in logic programs. Structured data items require a careful definition of the selection program semantics. If, besides r_1 as above, rule r_2 says that source s_2 is in general best for awards, then r_1 should have priority over r_2 and s_1 should be queried. Note that this priority is not based on inheritance (cf. [7]), but on the *context* of the occurrence of awards, which is determined by the access path in the query. Furthermore, implicit priorities, derived from context information, must be combined with explicit user preferences from the selection policy, and arising conflicts must be resolved.

Generally speaking, the main advantage of using extended logic programs under the answer set semantics for dealing with the site-selection task considered here is given by their declarative nature, which provides a clear semantics for knowledge representation and reasoning, both under qualitative as well as quantitative criteria. Furthermore, this formalism is capable of handling incomplete information, modeling some form of nonmonotonic reasoning which, arguably, is an inherent feature of the current problem domain. As well, changes in the specification of the site-selection process are easily incorporated by modifying or adding suitable rules or constraints, without the need for redesigning the given program, as might be the case, e.g., in procedural languages. Finally, the declarative nature of the answer set semantics formalism permits also a coupling with sophisticated ontology tools for providing more advanced features for the domain knowledge.

2 Outline of the Approach

Our approach is based on the *answer set programming* paradigm [28], in which problems are represented by extended logic programs (ELPs) [21], possibly augmented with priorities (e.g., [6, 11, 25]) and weak constraints [8], and solutions are obtained from the answer sets of the programs. We assume the reader familiar with answer sets; for further background, see e.g., [21, 28, 4].

Let us illustrate our conception with the following scenario.

Example 1 *Consider an information agent which knows information sources s_1, s_2, and s_3 about movies, containing XML data. Assume that the following query (in XML-QL [13]) is handed to the agent, which informally asks a source for titles of all movies directed by Alfred Hitchcock:*

```
FUNCTION HitchcockMovies($MovieDB:"Movie.dtd") {
  CONSTRUCT <MovieList> {
    WHERE <MovieDB> <Movie>
            <Title> $t </Title>
            <Director> <Personalia>
              <FirstName> "Alfred" </FirstName>
              <LastName> "Hitchcock" </LastName>
             </Personalia> </Director>
          </Movie> </MovieDB>
    IN source($MovieDB)
    CONSTRUCT <Movie> $t </Movie>}
  </MovieList>}
```

Suppose the agent knows that s_1 is especially accurate for information about directors, while s_2 has usually good coverage of information about person data; all which known about s_3 is that the site is not particularly reliable. In this situation, we would expect the agent to select s_1 for querying.

While the above example is simple, it shows that the selection process uses different kinds of knowledge. In our approach, this is reflected by the following components of a *selection base* $\mathcal{S} = \langle \Pi_{qd}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u \rangle$ (see Figure 1):

Query description Π_{qd}: For any query Q as in Example 1, a high-level abstract description is extracted from a low-level (syntactic) representation $R(Q)$, given as a set of elementary facts, by applying a logic program Π_{qd} to $R(Q)$. Informally, the rules of Π_{qd} single out the essential parts of Q, such as occurrence of attributes and values in the query, comparison and joins, or access paths describing contexts.

Domain theory Π_{dom}: The agent's background knowledge about the application domain (e.g., movies) is represented in a logic program Π_{dom}. It includes a domain ontology, which is established by the knowledge engineer; under certain circumstances, semi-automatic support may be provided from meta-information about the data stored in information sites (e.g., an XML DTD).

Site description Π_{sd}: Information about the sites is represented in a logic program Π_{sd}, using special predicates for (i) *thematic aspects* concerning query topics, e.g., $accurate(S,T,V)$, stating that site S is accurate on topic T at value $V \in \{high, medium, low\}$; (ii) *cost aspects*, e.g., $charge(S,C)$, expressing that querying site S has cost C; and (iii) *technical aspects*, e.g., $last\_update(S,D)$, which gives the date D of the last update of site S. Besides facts, Π_{sd} may contain also rules, and in particular default rules for incomplete information.

Site-selection program Π_{sel}: The selection of information sites is specified using rules and constraints, which refer to predicates defined in the above programs. It comprises both *qualitative aspects*, given by rules and constraints, as well as *quantitative aspects* in terms of optimization criteria (concerning, e.g., cost or response time), which are represented using weak constraints [8].

The user may also define preferences between rules, in terms of a strict partial order, $<_u$. These preferences must be combined with implicit priorities derived from the *context* in which rules for site selection should be applied. We define a respective semantics, which resolves also possible conflicts that might arise.

Summarizing, given a query Q, the overall evaluation proceeds in three steps:

Step 1 (query representation): The input query Q is parsed and mapped into the internal query representation $R(Q)$.

Step 2 (qualitative selection): From $R(Q)$, Π_{sd}, and Π_{dom}, the qualitative part of Π_{sel} is used to single out different query options by respecting qualitative aspects only, where explicit preferences, $<_u$, and implicit priorities must be taken into account. To this end, a priority relation $<$ is computed on rules, which is then used in a prioritized logic program $(\Pi_Q, <)$. Candidate solutions are computed as answer sets of $(\Pi_Q, <)$.

Step 3 (optimization): Among the candidates of Step 2, the one is chosen which represents the best solution with respect to the quantitative aspects of Π_{sel}, and the selected site is provided as output.

In this paper, we focus on Π_{sel} and the construction of $(\Pi_Q, <)$ (see Section 4). The next section provides some details concerning the query-description program Π_{qd}; a comprehensive discussion of the employed query abstraction method is reported in a companion paper [17].

3 Abstract Query Description

An important aspect of our approach is a meaningful description of a given query Q. For our purposes, we need a suitable representation of the constituents of Q in terms of predicates and objects. Simply mapping Q to logical facts reflecting its syntactic structure, which can be easily derived from the query grammar, will not serve our purposes. Rather, we need an *abstract* description which provides "relevant" information about Q, such as occurrence of an attribute or a value in Q, related to the scope in which it occurs. As an example, the value "Hitchcock" occurs in the query of Example 1 in a selection condition, which is performed on the attribute *LastName* reached by the access path *Movie/Director/Personalia*.

In our abstract query description, we adopt a general view in which a query consists of a *construct part*, a *where part*, and a *source part*, corresponding to the SQL *select*, *where*, and *from parts*, respectively. At a high-level description, the following predicates are used for describing a query Q, using surrogates for maximal access paths (starting from a root object) in Q from the low-level description:

$access(O,C,P,Q)$: This predicate describes that an item, which may be an attribute or a concept, is accessed under path P within the context of a concept C in the where-part of query Q, where O identifies a maximal path in Q with suffix "C/P". We call such a pair (C,P) a *context-reference pair in* Q.

query(Q): identifies an individual, independent query Q, taking into account that a query might be composed of several such subqueries.

occurs(O, V): this means that a value V is associated with a maximal path O in the (global) query. For example, *occurs*$(o_2, \text{"}Alfred\text{"})$ expresses that value *Alfred* is associated with the maximal path o_2.

selects(O, C, V): like *occurs*, but also specifies the type of relationship in terms of comparison operator C. For example, we have *selects*$(o_2, equal, \text{"}Alfred\text{"})$, where *equal* represents the equality operator.

constructs(O, I, P): specifies the maximal paths O which, by use of variables, also appear in the construct part of the query, as an item I under path P (which may be different from the path in the where part). In our example, we have *constructs*$(o_1, \text{"}Movie\text{"}, \text{" "})$.

joins(O_1, O_2, C): records joins of (or within) queries between maximal paths O_1 and O_2 under comparison operator C. Our example has no join.

Example 2 *The complete high-level description of the query in Example 1 is given by the following set of facts:*

$\{$ *query*(q_1),
access$(o_1, \text{"}MovieDB\text{"}, \text{"}Movie/Title\text{"}, q_1)$,
access$(o_1, \text{"}Movie\text{"}, \text{"}Title\text{"}, q_1)$,
access$(o_2, \text{"}MovieDB\text{"}, \text{"}Movie/Director/Personalia/FirstName\text{"}, q_1)$,
access$(o_2, \text{"}Movie\text{"}, \text{"}Director/Personalia/FirstName\text{"}, q_1)$,
access$(o_2, \text{"}Director\text{"}, \text{"}Personalia/FirstName\text{"}, q_1)$,
access$(o_2, \text{"}Person\text{"}, \text{"}FirstName\text{"}, q_1)$,
access$(o_3, \text{"}MovieDB\text{"}, \text{"}Movie/Director/Personalia/LastName\text{"}, q_1)$,
access$(o_3, \text{"}Movie\text{"}, \text{"}Director/Personalia/LastName\text{"}, q_1)$,
access$(o_3, \text{"}Director\text{"}, \text{"}Personalia/LastName\text{"}, q_1)$,
access$(o_3, \text{"}Person\text{"}, \text{"}LastName\text{"}, q_1)$,
occurs$(o_2, \text{"}Alfred\text{"})$, *occurs*$(o_3, \text{"}Hitchcock\text{"})$,
selects$(o_2, equal, \text{"}Alfred\text{"})$,
selects$(o_3, equal, \text{"}Hitchcock\text{"})$,
constructs$(o_1, \text{"}Movie\text{"}, \text{" "})$ $\}$

They are obtained from the maximal paths MovieDB/Movie/Title and MovieDB/Movie/Director/Personalia/FirstName corresponding to o_1 and o_2, respectively. Here, "MovieDB", "Movie", "Director", and "Person" are concepts, given in the ontology; furthermore, "Personalia" is known to amount to "Person".

Note that the high-level description of Q does not depend on a particular query language. It is extracted from a low-level query description, $R(Q)$, which contains further structural details and can be generated by a query parser, by means of a stratified logic program Π_{qd} which imports the relevant ontology knowledge. The high-level description facts are given by the (unique) answer set of $\Pi_{qd} \cup Q(R)$. Further details are provided in [17].

4 Site Selection

In this section, we describe the formal details of site-selection programs, which represent the main part of our architecture. Roughly speaking, a site-selection program Π_{sel} is a prioritized logic program consisting of four parts, namely (i) a core unit Π^c_{sel}, containing the actual site-selection rules, (ii) a set Π^{aux}_{sel} of auxiliary rules, (iii) an order relation $<_u$ defined over members of Π^c_{sel}, and (iv) an optimization part Π^o_{sel}, containing weak constraints. We define syntax and semantics of site-selection programs, and discuss some basic properties.

4.1 Syntax

The alphabet, $\mathcal{A}_{sel}$, of a site-selection program Π_{sel} in a selection base $\mathcal{S} = \langle \Pi_{qd}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u \rangle$ consists of the following pairwise disjoint categories:

- the alphabets $\mathcal{A}_{qd}$, $\mathcal{A}_{sd}$, and $\mathcal{A}_{dom}$ of the query description Π_{qd}, the site description Π_{sd}, and the domain theory Π_{dom}, respectively;
- the selection predicate *query_site*(S, Q), expressing that site S is selected for evaluating query Q; and the predicates *default_object*(O, C, Q) and *default_path*(O, P, Q), which are projections of *access*(O, C, P, Q);
- a set $\mathcal{A}_{aux}$ of auxiliary predicates; and
- a set N of terms serving as *names* for rules.

Informally, the predicates *default_object*(O, C, Q) and *default_path*(O, P, Q) serve to specify a default status for selection rules depending on context-reference pairs matched in the query Q. For example, *default_object*$(O, \text{"}Person\text{"}, Q)$ in the body of rule r says that r is eligible, if the concept *Person* occurs in

r_1: $query\_site(s_2, Q) \leftarrow default\_object(O, \text{"Person"}, Q);$

r_2: $query\_site(s_1, Q) \leftarrow selects(O, equal, \text{"Hitchcock"}),$
$access(O, \text{"Director"}, \text{"Personalia/LastName"}, Q);$

r_3: $query\_site(S, Q) \leftarrow default\_path(O, \text{"LastName"}, Q),$
$default\_object(O, T, Q),$
$accurate(S, T, high);$

r_4: $high\_acc(T, Q) \leftarrow access(O, T, P, Q),$
$accurate(S, T, high);$

r_5: $high\_cov(T, Q) \leftarrow access(O, T, P, Q),$
$covers(S, T, high);$

c_1: $\Leftarrow query\_site(S, Q),$
$high\_acc(T, Q),$
$not\ accurate(S, T, high)$ [10 : 1];

c_2: $\Leftarrow query\_site(S, Q),$
$high\_cov(T, Q),$
$not\ covers(S, T, high)$ [5 : 1];

$n_{r_1(Q,\_)} <_u n_{r_3(Q,\_,\_,\_)}.$

Figure 2: Site-Selection Program $(\Pi_{sel}, <_u)$ from Example 3.

the access path O and there is no other rule s that refers to some context-reference pair (C', P') matched in Q. These defaults are semantically realized using a suitable rule ordering.

We assume that $\mathcal{A}_{sel}$ generates a function-free language; rules are constructed as usual. We use upper-case letters for variables and lower-case letters for constants. We write $r(X_1, \ldots, X_n)$ to indicate that rule r has variables $X_1, \ldots, X_n$. As well, $H(r)$ denotes the head of r and $B(r)$ its body. The set of all literals built from the atoms in $\mathcal{A}_\ell$, where $\ell \in \{qd, sd, dom, aux, sel\}$, is denoted by Lit_ℓ. For literal L, we write $\neg L$ to denote its complementary literal, i.e., $\neg L = A$ if $L = \neg A$, and $\neg L = \neg A$ if $L = A$, for some atom A. Furthermore, we use an injective function $n(\cdot)$ which assigns to each rule $r(X_1, \ldots, X_n) \in \Pi_{sel}$ a name $n(r) \in N$ of form $t(X_1, \ldots, X_n)$, such that $n(\cdot)$ naturally extends to Π_{sel} by $n(s) = t(g_1, \ldots, g_n)$ for each ground instance $s = r(g_1, \ldots, g_n)$ of r. To ease notation, we also write n_r instead of $n(r)$.

Definition 1 *A site-selection program over $\mathcal{A}_{sel}$ is a tuple $(\Pi_{sel}, <_u)$, where*

(*i*) *Π_{sel} is a collection of rules over $\mathcal{A}_{sel}$ consisting of the following parts:*

(*a*) *the* core unit *Π^c_{sel} containing rules of form*

$$query\_site(S, Q) \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n;$$

(*b*) *a set Π^{aux}_{sel} of* auxiliary rules *of form*

$$L_0 \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n;$$

(*c*) *an* optimization part *Π^o_{sel} containing weak constraints* [8] *of form*

$$\Leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n\ [w : l],$$

where L_0 is either a literal from Lit_{aux} or is of form $\neg query\_site(\cdot, \cdot)$, and $L_i \in Lit_{sel}$ for $1 \leq i \leq n$, and $w, l \geq 1$ are integers. The number w is the weight *and l is the* priority level *of the weak constraint in (c).*

(*ii*) *The relation $<_u$ is a strict partial order (i.e., an irreflexive and transitive relation) between names of rules in Π^c_{sel}, whose elements are called* user defined preferences. *If $n_r <_u n_s$ then s is said to have* preference over *r.*

The rules in the core unit Π^c_{sel} serve for selecting a site, based on information from the domain description Π_{dom}, the site description Π_{sd}, the query description $R(Q) \cup \Pi_{qd}$, and possibly from auxiliary rules.

The latter may be used, e.g., for evaluating complex conditions. By $<_u$, preference of site selection can be expressed. The weak constraints in Π^o_{sel} are used to filter answer sets under *quantitative conditions*. Informally, they work as follows: From the answer sets of the weak-constraint free part of a program Π, prune first those where the sum of weights of violated constraints in the highest priority level is not minimal, then those where the sum of weights of violated constraints in the next lower level is not minimal, and so on. We refer the reader to [8] for a precise formal definition.

Example 3 *Consider some site-selection program $(\Pi_{sel}, <_u)$ for our movie domain, depicted in Figure 2. Intuitively, rule r_1 advises to choose site s_2 if the query involves persons and no more specific rule is eligible. Furthermore, rule r_2 says to choose site s_1 if the query contains an explicit select on the movie director Hitchcock. Rule r_3 demands to choose a site if, on some query access path, "LastName" is accessed under some concept T (with arbitrary intermediate access path), and the site is highly accurate for T. Rules r_4 and r_5 define auxiliary predicates which hold on concepts T appearing in the query Q such that some site with high accuracy and coverage, respectively, for T exists. The weak constraints c_1 and c_2 state penalties for choosing a site that has not high accuracy (weight 10) or coverage (weight 5), respectively, for a concept in the query while such a site exists. Finally, $n_{r_1(Q,\_)} <_u n_{r_3(Q,\_,\_,\_)}$ expresses that on the same query, instances of rule r_3 are preferred to those of r_1.*

4.2 Semantics

The semantics of a site-selection program $(\Pi_{sel}, <_u)$ in a selection base $\mathcal{S} = \langle \Pi_{qd}, \Pi_{dom}, \Pi_{sd}, \Pi_{sel}, <_u \rangle$ on a query Q is given by the concept of an *answer set* of $(\Pi_{sel}, <_u)$, which is defined as a preferred answer set of a prioritized logic program $\mathcal{E}(\mathcal{S}, Q) = (\Pi_Q, <)$ associated with $\mathcal{S}$ and Q. The program Π_Q contains ground instances of rules and constraints in Π_{sel}, and further rules ensuring that a single site is selected per query and defining the default context predicates. The order relation $<$ is formed from the user preferences $<_u$ and the implicit priorities derived from context-references in the core unit and from auxiliary rules. In that, we must suitably combine preference information—and in particular handle conflicts. For this, we use a cautious conflict elimination policy.

We commence the formal details with the following notation: For any rule r, its *defaultization*, r^Δ, is given by $H(r) \leftarrow B(r), not\ \neg H(r)$.

Definition 2 *Let $\mathcal{S} = \langle \Pi_{qd}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u \rangle$ be a selection base and Q a query. Then, the program Π_Q contains all ground instances of the rules and constraints in $\Pi^{aux}_{sel} \cup \Pi^o_{sel}$, as well as all ground instances of the following rules:*

1. *the defaultization r^Δ of r, for each $r \in \Pi^c_{sel}$;*
2. *the* structural rule

$$\neg query\_site(S, Q) \leftarrow query\_site(S', Q), S \neq S';$$

and

3. *the* default context rules

$$default\_object(O, T, Q) \leftarrow access(O, T, \_, Q);$$
$$default\_path(O, P, Q) \leftarrow access(O, \_, P, Q).$$

Intuitively, the defaultization makes the selection rules in Π_{sel} defeasible; the structural rule enforces that only one source is selected; and the default context rules define the two default predicates. We remark that, since we have no function symbols in our language, Π_Q is finite, and its size depends on the constants appearing in Π_{qd}, $R(Q)$, Π_{sd}, and Π_{dom}.

For $\mathcal{S} = \langle \Pi_{qd}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u \rangle$ and query Q, we call any answer set of $\Pi = \Pi_{qd} \cup R(Q) \cup \Pi_{sd} \cup \Pi_{dom}$ a *selection input of $\mathcal{S}$ for Q*. The set of all selection inputs of $\mathcal{S}$ for Q is denoted by $Sel(\mathcal{S}, Q)$. For $I \in Sel(\mathcal{S}, Q)$, we define

$$I_{def} = I \cup \{default\_object(o, c, q) \mid access(o, c, p, q) \in I\} \cup \{default\_path(o, p, q) \mid access(o, c, p, q) \in I\}.$$

Notice that, in general, a selection base may admit multiple selection inputs for a query Q. However, in many cases there may exist only a single selection input, in particular, if the site description Π_{sd} and the domain knowledge Π_{dom} have unique answer sets. In our framework, this is ensured if, for instance, these components are represented by stratified programs.

Definition 3 *A rule $r \in \Pi_Q$ is* relevant *for Q iff there is some $I \in Sel(\mathcal{S}, Q)$ such that $B^\dagger(r)$ is true in I_{def}, where $B^\dagger(r)$ results from $B(r)$ by deleting each element which does not contain a predicate symbol from $\mathcal{A}_{qd} \cup \mathcal{A}_{sd} \cup \mathcal{A}_{dom} \cup \{default\_object, default\_path\}$.*

In the sequel, we denote for any binary relation R its transitive closure by R^*.

We continue with the construction of the preference relation $<$, used for interpreting a site-selection program

$(\Pi_{sel}, <_u)$ relative to a selection base $\mathcal{S}$ and a query Q, in terms of an associated prioritized logic program $(\Pi_Q, <)$.

Informally, the specification of $<$ depends on the following auxiliary relations:

- the preference relation $\preceq_c$, taking care of implicit context priorities;
- the intermediate relation $\trianglelefteq$, representing a direct combination of user-defined preferences with context preferences; and
- the preference relation $<'$, removing possible conflicts within the joined relation $\trianglelefteq$ and ensuring transitivity of the resultant order $<$.

More specifically, the relation $\preceq_c$ represents the first step in the construction of $<$, transforming structural context information into explicit preferences, in virtue of the following specificity conditions:

- relative to the same maximal path and query, rules that mention default concepts or a context-reference pair matching the query are considered more specific than rules mentioning default paths;
- default contexts for concepts are assumed to be more specific than default contexts for attributes; and
- rules which have in the same maximal path more detailed context-reference pairs are considered more specific than rules with less such objects.

The second step in the construction of $<$ is the relation $\trianglelefteq$, which is just the union of the user preferences $<_u$ and the context priorities $\preceq_c$. In general, this will not be a strict partial order. To enforce irreflexivity, we remove all priorities $n_r \trianglelefteq n_s$ which lie on a cycle $n_r \trianglelefteq n_s \trianglelefteq n_{s_1} \trianglelefteq \cdots \trianglelefteq n_{s_n} \trianglelefteq n_r$, resulting in $<'$. Finally, taking the transitive closure of $<'$ yields $<$. The formal definition of relation $<$ is as follows.

Definition 4 *Let $\mathcal{S}$ be a selection base, Q a query, and Π_Q as in Definition 2. We introduce the following binary relations over the names of rules in Π_Q: For $r, s \in \Pi_Q$,*

1. *$n_r \preceq_c n_s$ iff r, s are relevant for Q, $r \neq s$, and one of (P_1)-(P_3) holds:*
   - (P_1) *default_path(o_1, p_1, q) is a member of $B(r)$, and we have either default_object$(o_2, t_2, q) \in B(s)$ or access$(o_2, t_2, p_2, q) \in B(s)$;*
   - (P_2) *default_object(o_1, t_1, q) belongs to $B(r)$ and access(o_2, t_2, p_2, q) is in $B(s)$;*
   - (P_3) *access(o, t_1, p_1, q) is a member of $B(r)$, access(o, t_2, p_2, q) is in $B(s)$, and t_1/p_1 is a subpath of t_2/p_2;*
2. *$n_r \trianglelefteq n_s$ iff $n_r <_u n_s$ and r, s are relevant for Q, or $n_r \preceq_c n_s$;*
3. *$n_r <' n_s$ iff $n_r \trianglelefteq n_s$ but not $n_s \trianglelefteq^* n_r$.*

Then, the relation $<$ is given as the transitive closure of relation $<'$.

Let us illustrate this definition with an example.

Example 4 *Reconsider $(\Pi_{sel}, <_u)$ from Example 3. Suppose the domain ontology contains the concepts "MovieDB", "Actor", "Movie", "Director", and "Person", and that "Personalia" amounts to "Person". Furthermore, let us assume we have a unique selection input I for the query Q of Example 1, containing the following facts from the site description:*

$$accurate(s_1, \text{"}Director\text{"}, high);$$
$$covers(s_2, \text{"}Person\text{"}, high);$$
$$reliable(s_3, low),$$

as well as the following elements resulting from the query description and the default context rules:

$$default\_path(o_3, \text{"}LastName\text{"}, q_1);$$
$$default\_object(o_2, \text{"}Person\text{"}, q_1);$$
$$default\_object(o_3, \text{"}Person\text{"}, q_1);$$
$$default\_object(o_3, \text{"}Director\text{"}, q_1);$$
$$selects(o_3, equal, \text{"}Hitchcock\text{"});$$
$$access(o_2, \text{"}Person\text{"}, \text{"}FirstName\text{"}, q_1);$$
$$access(o_2, \text{"}Director\text{"}, \text{"}Personalia/FirstName\text{"}, q_1);$$
$$access(o_3, \text{"}Person\text{"}, \text{"}LastName\text{"}, q_1);$$
$$access(o_3, \text{"}Director\text{"}, \text{"}Personalia/LastName\text{"}, q_1).$$

These elements are exactly those contributing to relevant instances of Π_Q. The relevant instances of r_1, r_2, and r_3 are given by the ground rules $r_1(q_1, o_2)$, $r_2(q_1, o_3)$, $r_2(q_1, o_3)$, and $r_3(q_1, s_1, o_3, \text{"}D\text{"})$ (for brevity, we write here "D" for "Director"). Intuitively, we expect $r_2(q_1, o_3)$ to have highest priority among these rule instances, since the bodies of the instances of r_1 and r_3 contain default predicates while r_2 references a specific context. Actually, the order relation $<$ includes for the relevant instances of r_1, r_2, and r_3 the following pairs:

$$n_{r_1(q_1,o_2)} < n_{r_2(q_1,o_3)};$$

$n_{r_1(q_1,o_3)} < n_{r_2(q_1,o_3)}$;
$n_{r_3(q_1,s_1,o_3,\text{"}D\text{"})} < n_{r_2(q_1,o_3)}$.

Note that for each of r_4 and r_5, there also exist two relevant instances. However, they have no influence on the above rule ordering. Informally, they are either unrelated to or "ranked between" the priority of $r_2(q_1,o_3)$ and the priority of the relevant instances of r_1 and r_3 (since the 'access' predicates of r_4 and r_5 refer to the same context as the context referenced in the body of r_2, or to a subpath of such a context). Hence, the relevant instance of r_2 has highest priority.

With respect to r_1 and r_3, the auxiliary relation $\trianglelefteq$ contains two further structural priorities, namely

$$n_{r_3(q_1,s_1,o_3,\text{"}D\text{"})} \preceq_c n_{r_1(q_1,o_2)}$$

and

$$n_{r_3(q_1,s_1,o_3,\text{"}D\text{"})} \preceq_c n_{r_1(q_1,o_3)}.$$

They are in conflict with the user preferences

$$n_{r_1(q_1,o_2)} <_u n_{r_3(q_1,s_1,o_3,\text{"}D\text{"})}$$

and

$$n_{r_1(q_1,o_3)} <_u n_{r_3(q_1,s_1,o_3,\text{"}D\text{"})},$$

respectively. This conflict is resolved in the resultant relation $<$ by removing these preferences.

Note that the final order $<$ in Definition 4 enforces a cautious conflict resolution strategy, in the sense that it remains "agnostic" with respect to priority information causing conflicts. Alternative definitions of $<'$, such as removal of a minimal cutset eliminating all cycles in $\trianglelefteq$, may be considered; however, this may lead to nondeterministic choices. Namely, in general, multiple such cutsets exist, of which one must be chosen. Different choices lead to different orders $<$, which may lead to different results of the site selection program. Thus, unless a well-defined *particular* minimal cutset is singled out, by virtue of preference conflicts the result of the site selection process might not be deterministic. Furthermore, an extended logic program component computing a final order based on minimal cutsets is more involved than a component computing the relations in Definition 4.

Combining Definitions 2 and 4, we obtain the translation $\mathcal{E}(\cdot,\cdot)$ as follows:

Definition 5 *Let S be a selection base and Q a query. Then, the evaluation $\mathcal{E}(S,Q)$ of S with respect to Q is given by the prioritized logic program $(\Pi_Q, <)$, where Π_Q and $<$ are as in Definitions 2 and 4, respectively.*

For defining answer sets of site-selection programs, in what follows we assume the approach of [11] as underlying preference framework; other approaches, like, e.g., [6], are also suitable for this purpose. Furthermore, for any program Π and any set S of literals, we write $\Pi \cup S$ to denote the program $\Pi \cup \{L \leftarrow \mid L \in S\}$.

Definition 6 *Let $S = (\Pi_{qd}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u)$ be a selection base, and let $\mathcal{E}(S,Q) = (\Pi_Q, <)$ for query Q. Then, $S \subseteq Lit_{sel}$ is an* answer set of $(\Pi_{sel}, <_u)$ for Q with respect to S *iff S is a preferred answer set of the prioritized logic program $(\Pi_Q \cup I, <)$, for some $I \in Sel(S,Q)$.*

A site s is selected for Q *iff the atom $query\_site(s,q)$ belongs to some answer set of $(\Pi_{sel}, <_u)$ for Q (with respect to S), where the constant q represents Q.*

Example 5 *In our running example, $(\Pi_{sel}, <_u)$ has a unique answer set, S, for the query Q from Example 3 with respect to S, containing $query\_site(s_1,q_1)$ (where q_1 represents Q). This atom is derived from the core rule $r_2(q_1,o_3)$, which has the highest priority among the applicable rules, leading to a single preferred answer set for the weak-constraint free part of Π_{sel}. If we replace, e.g., r_1 by the rule*

$$query\_site(s_2,Q) \leftarrow access(O,\text{"}Person\text{"},P,Q)$$

and adapt the corresponding user preference to

$$n_{r_1(Q,\_,\_)} <_u n_{r_3(Q,\_,\_,\_)},$$

then the weak-constraint free part of Π_{sel} has two preferred answer sets: one, S_1, is identical to S (where an application of $r_2(q_1,o_3)$ is preferred to an application of $r_1(q_1,o_3)$ given that $n_{r_1(q_1,o_3)} < n_{r_2(q_1,o_3)}$); in the other answer set, S_2, the rule $r_1(q_1,o_2)$ is applied and $query\_site(s_2,q_1)$ is derived. Informally, due to the replacement, the preference of $r_2(q_1,o_3)$ over $r_1(q_1,o_2)$ does no longer hold, since their 'access' predicates refer to different contexts (".../FirstName" and ".../LastName", respectively). As a consequence, $r_1(q_1,o_2)$ has maximal preference like $r_2(q_1,o_3)$.

Given that S_1 has weight 5, caused by violation of $c_2(s_1,q_1,\text{"}Person\text{"})$, but S_2 has weight 10, caused by violation of $c_1(s_2,q_1,\text{"}Director\text{"})$, S_1 is the answer set of $(\Pi_{sel}, <_u)$ for Q.

4.3 Properties

Finally, we discuss some properties of our framework. Our first result deals with the adequacy of the evaluation method of site-selection programs with respect to the usual semantics of prioritized logic programs.

Recall that the construction of answer sets of site-selection programs is laid out in a modular fashion, depending on the input of several subprograms, where each program module is assigned with a particular representation task. However, since the predicate symbols occurring in the heads of rules in a site-selection program do not occur in rules from the query description, the site description, or the domain theory, from standard modularity results for logic programs [27] we easily obtain the following characterization:

Theorem 1 *Let $S = \langle \Pi_{qd}, \Pi_{dom}, \Pi_{sd}, \Pi_{sel}, <_u \rangle$ be a selection base, let Q be a query, and let $\mathcal{E}(S, Q) = (\Pi_Q, <)$. Then, S is an answer set of $(\Pi_{sel}, <_u)$ for Q with respect to S iff S is a preferred answer set of $(\Pi_Q \cup \Pi_{qd} \cup R(Q) \cup \Pi_{sd} \cup \Pi_{dom}, <)$.*

Furthermore, it is possible to emulate the construction of $\mathcal{E}(S, Q)$ in terms of a *single* logic program under usage of dynamic priorities. Indeed, given the relevant rules for Q, the order relation $<$ can be computed by a stratified logic program $\Pi_<$ which computes the relations in Definition 4. On the other hand, the relevant rules for Q can be computed in a program Π_{rel} derived from $\Pi_{qd} \cup \Pi_{sd} \cup \Pi_{dom}$ using weak constraints. By combining the components, we obtain:

Theorem 2 *Let $S = \langle \Pi_{qd}, \Pi_{dom}, \Pi_{sd}, \Pi_{sel}, <_u \rangle$ be a selection base. Then, there exists a logic program $\Pi_S = \Pi_{qd} \cup \Pi_{dom} \cup \Pi_{sd} \cup \Pi_{sel} \cup <_u \cup \Pi_< \cup \Pi_{rel}$, such that for every query Q, the preferred answer sets of $\Pi_S \cup R(Q)$ under dynamic preference $<$ correspond to the answer sets of $(\Pi_{sel}, <_u)$ for Q with respect to S.*

One of the desiderata of our approach is that each answer set selects a unique source, for any query Q. The following result states that this property is fulfilled.

Theorem 3 *Let S be an answer set of $(\Pi_{sel}, <_u)$ for query Q with respect to S. Then, for any constant q it holds that*

$$|\{s \mid query\_site(q, s) \in S\}| \leq 1.$$

Intuitively, this result holds due to the presence of the structural rule

$$\neg query\_site(S, Q) \leftarrow query\_site(S', Q), S \neq S'$$

in Definition 2, which enforces that during the evaluation step no more than one site can be selected simultaneously.

Lastly, the following result concerns the order of application of site-selection rules, stating that site selection is blocked in terms of priorities as desired.

Theorem 4 *Let S be an answer set of $(\Pi_{sel}, <_u)$ for query Q with respect to S, and let r be some rule from the grounding of Π^c_{sel} for Q with respect to S. Suppose that $B(r)$ is true in S but $H(r) \notin S$. Then, there is some s in the grounding of $\Pi^c_{sel} \cup \Pi^{aux}_{sel}$ for Q with respect to S such that*

1. *$B(s)$ is true in S;*
2. *$H(s) \in S$; and*
3. *either r and s are incompatible with respect to $<$, or else $r < s$ holds.*

This result follows from the observation that, under the above circumstances, if $B(r)$ is true in S but $H(r) \notin S$, then there must be some s in the grounding of $\Pi^c_{sel} \cup \Pi^{aux}_{sel}$ such that $H(s) = \neg H(r)$ and s fires during the construction of S. Moreover, the underlying preference semantics guarantees that we can choose s in such a way that either r and s are not related with respect to $<$, or else $r < s$ holds.

5 Related Work and Conclusion

Selection of data sources is an ingredient to several systems for information integration, e.g. [2, 5, 9, 20, 22, 24, 26]. However, they center around mappings from a global scheme to local schemes and vice versa, query rewriting, and planning, in order to optimally reconstruct dispersed information. Our concern is with the *qualitative* selection from different alternatives, based on rich meta-knowledge, which is not an issue there. More related is [23], which outlines an interactive tool for assisting information specialists in query design. It relieves them from searching through data source specifications and can provide suggestions for using sources to determine tradeoffs. However, no formal semantics or richer domain theories, capable of handling incomplete and default information, is presented. Remotely related to our work is [19], which presents a decision-theoretic model for selecting data sources based on retrieval cost and typical IR parameters.

Our approach to declarative information site selection, based on query analysis and contexts in logic programs, is novel and innovate in several respects. It is implemented on top of the `dlv` system [18, 15] and its front end `plp` [12] for prioritized logic programs, and employs the underlying Eclipse Prolog engine as a glueing frame. For testing, we developed a prototypical environment of a movie domain, which comprises (i) basic domain knowledge, (ii) XML sources containing movie data wrapped from the Internet Movie Database [1] and other movie related data sources, and

(iii) suitable site descriptions. Queries are formulated in XML-QL [13], and can be executed after site selection on the respective source. For generating the low-level representation $R(Q)$, for query Q, we have developed an XML-QL query parser, written in C++ using the standard tools `lex` and `yacc`. The implementation, as well as the full movie application, is too complex to be described here. In its current version, selection inputs are assumed to be unique, which is ensured if the site description Π_{sd} and the domain knowledge Π_{dom} yield unique answer sets. This is given, for example, if these components are stratified, or by adding suitable weak constraints. The implementation will be considered in more detail in an extended paper.

Our ongoing work comprises several tasks. One is making our site selection method available for building multi-agent information systems. To this end, its *agentization* in the *IMPACT* agent platform [3] is in progress. Another task is coupling the approach with learning and program updates for adaptive source selection, and with query rewriting and planning. Also, the current setting of single site selection can be generalized to multiple site selection for parallel querying.

References

[1] The Internet Movie Database. `http://imdb.com`.

[2] Y. Arens, C. Chee, C. Hsu, and C. Knoblock. Retrieving and Integrating Data from Multiple Information Sources. *Int. J. of Cooperative Information Systems*, 2(2):127–158, 1993.

[3] K. Arisha, T. Eiter, S. Kraus, F. Ozcan, R. Ross, and V. Subrahmanian. IMPACT: A Platform for Collaborating Agents. *IEEE Intelligent Systems*, 14(2):64–72, 1999.

[4] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving with Answer Sets.* 2001. Draft. Available from `http://www.public.asu.edu/~cbaral/bahi/`.

[5] R. Bayardo, B. Bohrer, R. Brice, A. Cichocki, J. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. InfoSleuth: Semantic Integration of Information in Open and Dynamic Environments (Experience Paper). In *SIGMOD Conf. 1997*, pages 195–206, 1997.

[6] G. Brewka and T. Eiter. Preferred Answer Sets for Extended Logic Programs. *Artificial Intelligence*, 109(1–2):297–356, 1999.

[7] F. Buccafurri, N. Leone, and P. Rullo. Stable Models and their Computation for Logic Programming with Inheritance and True Negation. *J. of Logic Programming*, 27(1):5–43, 1996.

[8] F. Buccafurri, N. Leone, and P. Rullo. Enhancing Disjunctive Datalog by Constraints. *IEEE Trans. on Knowledge and Data Engineering*, 12(5):845–860, 2000.

[9] P. Cannata, M. Huhns, N. Jacobs, T. Ksiezyk, K. Ong, A. Sheth, M. Singh, C. Tomlinson, and D. Woelk. The Carnot Heterogeneous Database Project: Implemented Applications. *Distributed and Parallel Databases*, 5(2):207–225, 1997.

[10] K. Decker, K. Sycara, and M. Williamson. Middle-Agents for the Internet. In *Proc. 15th Int. Joint Conf. on Artificial Intelligence (IJCAI'97)*, volume 1, pages 578–583. Morgan Kaufmann, 1997.

[11] J. Delgrande, T. Schaub, and H. Tompits. Logic Programs with Compiled Preference. In W. Horn, editor, *Proc. 14th Europ. Conf. on Artificial Intelligence (ECAI 2000)*, pages 392–398. IOS Press, 2000.

[12] J. Delgrande, T. Schaub, and H. Tompits. plp: A Generic Compiler for Ordered Logic Programs. In T. Eiter, W. Faber, and M. Truszczyński, editors, *Proc. 6th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR'01)*, pages 411–415. Springer, 2001.

[13] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. A Query Language for XML. In *Proc. 8th International World Wide Web Conference (WWW8)*, 1999. Computer Networks 31(11-16): 1155-1169. See also `http://www.w3.org/TR/NOTE-xml-ql/`.

[14] Y. Dimopoulos and A. Kakas. Information Integration and Computational Logic. *Computational Logic, Special Issue on the Future Technological Roadmap of Compulog-Net*, 2000. Available from `http://www.compulog.org/net/Forum/Supportdocs.html`.

[15] T. Eiter, W. Faber, N. Leone, and G. Pfeifer. Declarative Problem-Solving Using the DLV System. In J. Minker, editor, *Logic-Based Artificial Intelligence*, pages 79–103. Kluwer Academic Publishers, 2000.

[16] T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. Using Methods of Declarative Logic Programming

for Intelligent Information Agents. *Theory and Practice of Logic Programming*, 2001. To appear. Available from `http://xxx.lanl.gov/abs/cs.MA/0108008`.

[17] T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. Declarative Query Abstraction for Selecting Information Sources. Manuscript (in preparation), 2002.

[18] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. The KR System dlv: Progress Report, Comparisons, and Benchmarks. In A. Cohn, L. Schubert, and S. Shapiro, editors, *Proceedings Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR-98)*, pages 406–417, June 2–4 1998.

[19] N. Fuhr. A Decision-Theoretic Approach to Database Selection in Networked IR. *ACM Transactions on Information Systems*, 17(3):229–249, 1999.

[20] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, and J. Widom. The TSIMMIS Approach to Mediation: Data Models and Languages. *J. of Intelligent Information Systems*, 8(2):117–132, 1997.

[21] M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9(3–4):365–386, 1991.

[22] M. Genesereth, A. Keller, and O. Duschka. Infomaster: An Information Integration System. In *Proc. ACM SIGMOD Conference*, volume 26(2) of *SIGMOD Record*, pages 539–542, New York, 1997. ACM Press.

[23] S. B. Huffman and D. Steier. A Navigation Assistant for Data Source Selection and Integration. In *Working Notes of AAAI-95 Fall Symposium Series on AI Applications in Knowledge Navigation and Retrieval, Cambridge, MA*, pages 72–77, 1995.

[24] M. Huhns and M. Singh. The Semantic Integration of Information Models. In *AAAI Workshop on Cooperation among Heterogeneous Intelligent Agents*, 1992.

[25] K. Inoue and C. Sakama. Prioritized Logic Programming and Its Applications to Commonsense Reasoning. *Artificial Intelligence*, 123(1–2):185–222, 2000.

[26] A. Levy, A. Rajaraman, and J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In T. Vijayaraman, A. Buchmann, C. Mohan, and N. Sarda, editors, *Proc. 22th Int. Conf. on Very Large Data Bases (VLDB'96)*, pages 251–262. Morgan Kaufmann, 1996.

[27] V. Lifschitz and H. Turner. Splitting a Logic Program. In *Proceedings ICLP-94*, pages 23–38, Santa Margherita Ligure, Italy, June 1994. MIT-Press.

[28] A. Provetti and T. C. Son, editors. *Proceedings AAAI 2001 Spring Symposium on Answer Set Programming: Towards Efficient and Scalable Knowledge Representation and Reasoning, Stanford, CA (Workshop Technical Report SS-01-01)*. AAAI Press, March 2001.

[29] F. Sadri and F. Toni. Computational Logic and Multi-Agent Systems: a Roadmap. *Computational Logic, Special Issue on the Future Technological Roadmap of Compulog-Net*, 2000. Avaliable from `http://www.compulog.org/net/Forum/Supportdocs.html`.

Declarative & Procedural Goals in Intelligent Agent Systems

Michael Winikoff
School of Computer Science and Information Technology
RMIT University
Melbourne, Australia.
winikoff@cs.rmit.edu.au

Lin Padgham
School of Computer Science and Information Technology
RMIT University
Melbourne, Australia.
linpa@cs.rmit.edu.au

James Harland
School of Computer Science and Information Technology
RMIT University
Melbourne, Australia.
jah@cs.rmit.edu.au

John Thangarajah
School of Computer Science and Information Technology
RMIT University
Melbourne, Australia.
johnt@cs.rmit.edu.au

Abstract

An important concept for intelligent agent systems is *goals*. Goals have two aspects: declarative (a description of the state sought), and procedural (a set of plans for achieving the goal). A declarative view of goals is necessary in order to reason about important properties of goals, while a procedural view of goals is necessary to ensure that goals can be achieved efficiently in dynamic environments. In this paper we propose a framework for goals which integrates both views. We discuss the requisite properties of goals and the link between the declarative and procedural aspects, then derive a formal semantics which has these properties. We present a high-level plan notation with goals and give its formal semantics. We then show how the use of declarative information permits reasoning (such as the detection and resolution of conflicts) to be performed on goals.

1 Introduction

Intelligent agents are an important technology. Based on foundational work in AI and Philosophy, agent technology has significant applications in a wide range of domains (Jennings and Wooldridge, 1998) and are considered by some to be a natural successor to object oriented programming (Jennings, 2001). Although there has been much debate on what constitutes an agent, and which features are important, the consensus is that an *intelligent agent* is situated, autonomous, reactive, pro-active, and social (Wooldridge, 1998).

A clearly central concept for pro-active agents is that of *goals* (Winikoff et al., 2001). Goals have two aspects: *declarative*, where a goal is a description s of the state of the world which is sought ($Env \models s$); and *procedural*, where a goal is a set of procedures P which are executed (in an attempt) to achieve the goal[1]. Although the declarative aspect is perhaps more natural, the procedural is important for realisable agents: an important property of goals is that the agent have the capability of bringing about the goal, for example having the goal to make it stop raining isn't sensible, since (presumably) the weather isn't under the control of the agent (Padgham and Lambrix, 2000). Providing explicitly a procedural aspect which specifies how the agent might bring about the desired goal is one way of ensuring that the agent doesn't adopt goals which it has no way of bringing about.

Intelligent agent implementation platforms (such as PRS (Ingrand et al., 1992), dMARS (d'Inverno et al., 1998), JAM (Huber, 1999), JACK (Busetta et al., 1998), 3APL (Hindriks et al., 1999), and ConGOLOG (Giacomo et al., 2000)) are intended for deployment in highly dynamic environments and as a result adopt the procedural view of goals in order to avoid lengthy deliberation. For example, systems in the BDI (Belief Desire Intention) family treat goals as events which trigger plans, and 3APL defines a goal as being simply a procedure.

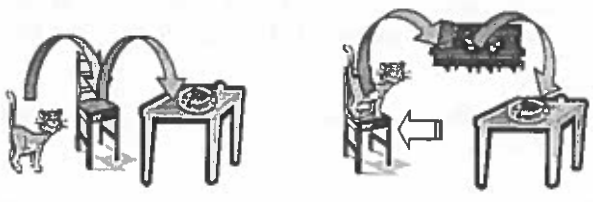
The use of the procedural aspect of goals is crucial to the practicality of these systems in highly dynamic environments. *However, by omitting the declarative aspect of goals the ability to reason about goals is lost.* Without knowing what a goal is trying to achieve, one cannot check whether the goal has been achieved, check whether the goal is impossible, or check for interference between goals (Thangarajah et al., 2002a). This lack of intelligence also constitutes a gap between BDI theories and implementations: an ideal BDI agent is required to drop goals when they are either achieved, or become unachievable (Rao and Georgeff, 1992). This cannot be done without declarative information. Further, declarative information allows the correct realisation of commitments to goals in BDI systems by decoupling plan failure from goal failure. A goal

[1] There is often more than one means of achieving a goal.

should not be dropped merely because a plan to achieve it has failed. Declarative information can specify a condition for dropping the goal which is independent from plan failure. By controlling our selection of goal discharge condition we can realise different commitment strategies.

We seek to develop a representation of goals which allows for both declarative and procedural aspects to be specified and used. Our aim is to allow agent implementation platforms to be more faithful to their theoretical foundations, and to provide better handling of goals. In doing this, it is important that the execution semantics for goals is consistent with the desired properties of (declarative) goals. For example, given a goal to achieve condition s using a set of procedures (or recipes or plans) P, if s holds, then P should not be executed.

We now present a simple example which illustrates the use of goals and the importance of the properties discussed. Consider a hungry cat. The cat knows there is food on the table and formulates a plan to jump on a chair, and then jump from the chair to the table (left diagram below). The cat leaps on to the chair successfully. At this point, a nearby human moves the chair so that the cat can no longer leap to the table. The cat (being an intelligent agent[2]) realises this, and revises its plans: the new plan is to leap from the chair to a shelf, then walk along the shelf, and finally leap from the shelf down to the table (right diagram below). However, fortune smiles upon the feline: after leaping up to the shelf it finds that food has been left on the shelf. The opportunistic cat abandons its plan to continue to the table and eats upon the shelf instead. This scenario illustrates the importance of retrying upon the failure of a plan, and of detecting when a goal (reaching food) is fortuitously achieved and dropping the goal and the associated plan. Another important property is that goals are not dropped when an associated plan succeeds, unless the goal's condition is met. For example, suppose the nearby human had moved the food rather than the chair. Our cat would then successfully perform it's original plan and reach the table, only to find that there was no food on the table. Although the plan succeeded, the goal of reaching food has not been achieved.



Seeking food involves the goal[3] Goal(atfood,*findfood*,nofood) where *findfood* is a set of plans for locating food, planning a path to reach the food, and following the path and where atfood $\equiv \exists X.\text{foodat}(X) \wedge \text{location}(X)$ and nofood $\equiv \neg\exists X.\text{foodat}(X)$.

The correct behaviour of the cat relies on goal execution having certain desired properties. In the next section we explore what these properties are and how the execution mechanism for goals can be derived so that it possesses these properties. In section 3 we present a plan notation with goals and give its formal semantics. Section 4 discusses how interaction between goals (both positive and negative - i.e. conflicts) can be reasoned about and managed by using declarative information.

2 Representing Goals

What properties should goals have? In specifying a goal construct this question is of obvious importance and we shall begin by looking at desired properties of goals. After identifying the desired properties of goals, we then derive a procedural interpretation for goals which meets these desired properties.

The BDI work of Rao and Georgeff (Rao and Georgeff, 1992) requires that successful and impossible attitudes are dropped[4] (that is, goals must not already be achieved and they must be possible), that goals are known to the agent (axiom *AI5* (Rao and Georgeff, 1991; Rao and Georgeff, 1998)) and that the set of goals held by an agent be consistent (Rao and Georgeff, 1992). Although persistence is left up to the commitment strategy, rather than being a property of the framework, the common commitment strategies require persistency of goals.

The formalisation of (van Linder et al., 1995) defines goals in terms of preferences. Preferences are persistent and known. Goals are selected from those preferences which are unfulfilled and realisable. Goals are also required to be consistent, so that where GA denotes that A is a goal, we have that (G$A \Rightarrow \neg$G$\neg A$).

The GOAL language (Hindriks et al., 2001) requires that goals not be entailed by beliefs (i.e. that they be unachieved) and that goals be satisfiable. The set of goals $\mathcal{G}$ is *not* required to be consistent: $\mathcal{G}$ can contain both p and $\neg p$. This is handled by not requiring that all goals be achieved simultaneously, so that if $\mathcal{G} = \{p, \neg p\}$ then p can be achieved and discharged and $\neg p$ can be achieved at a later time. A consequence of this is that the goal G$a \wedge b$ is not implied by the two goals Ga and Gb (since a and

[2] Or a robot: *www.necoro.com*

[3] The notation Goal(s, P, f) is read as "achieve condition s using the set of plans P; failing if f becomes true".

[4] The abstract BDI interpreter on page 441 includes the steps drop-successful-attitudes(B,G,I) and drop-impossible-attitudes(B,G,I) which correspond to an *open minded* commitment strategy.

b could be achieved at different times). Further, the goal operator does not distribute over implication.

Table 1: Properties of goals

	BDI	van Linder et al.	GOAL
Drop successful	✓	✓	✓
Drop impossible	✓	✓	✓
Known	✓	✓	
Consistent	✓	✓	
Persistent	(✓)	✓	

We thus require goals held by a rational agent to be *persistent*, *unachieved*, *possible*, *consistent*, and *known*:

Persistent: A goal should *only* be deleted when it succeeds or where there is a good reason for dropping it. A rational agent should not drop its goals without a good reason.

Unachieved: A goal to achieve s should be dropped when s is true. Note that this focuses on the desired outcome rather than the process - if we are in the middle of executing a plan P in order to bring about s and s becomes true, then the goal is dropped (with success) and the plan P aborted. A corollary is that a goal to achieve s where s already holds should trivially succeed without doing anything.

Possible: We specify a failure condition f which defines when a goal should be dropped with failure. In the same way that specifying s decouples the success of a goal from the success of its plans, specifying f decouples goal failure from plan failure. One advantage of allowing f to be specified is that it allows for a range of commitment strategies to be realised by specifying appropriate conditions for a goal to be dropped. We assume open minded commitment (Rao and Georgeff, 1992) as a default; in this case a goal is dropped with failure when it becomes *impossible* to achieve, i.e. f corresponds to an impossibility condition. We thus describe this desired property (goals are dropped with failure when f becomes true) as *possible*.

Consistent: A rational agent's goals are required to be *consistent*; an agent should not simultaneously pursue goals that conflict. For example, if an agent has a goal of eating the food on the table and another goal of eating food on the shelf at the same time, then the agent should recognise the conflict and resolve it in favour of the more important goal.

Known: Finally, a rational agent should know what goals it has; that is, goals should be *known*. This is an essential prerequisite to being able to reason about interactions between goals (see section 4).

Based on these properties we propose the construct $\mathsf{Goal}(s,P,f)$ which is read as "achieve s using P; failing if f becomes true" and which is viewed as being an executable statement in a plan language. Both s and f are logical formulae over about the agent's beliefs. We assume that s and f are exclusive, in that there is no world state S such that $S \models s \wedge f$. The procedural aspect, P, is a *set of guarded plans* of the form[5] $\langle\!\langle C_1 : P_1, \ldots, C_n : P_n \rangle\!\rangle$ where each C_i is a condition and each P_i is a plan designed to achieve s. If the condition C_i is true, then P_i may be executed; if there is more than one executable plan (i.e. more than one condition is true) than a non-deterministic choice is made between them.

The guiding intuition is that given an initial state of the world S_0, pursuing a goal $\mathsf{Goal}(s,P,f)$ will result in a (possibly infinite) sequence of states $\mathcal{S} = \langle S_0, S_1, S_2, \ldots \rangle$. Note that this sequence of states, which includes environmental changes, cannot be wholly determined by $\mathsf{Goal}(s,P,f)$ and S. However, it is clearly constrained by $\mathsf{Goal}(s,P,f)$, and it is the nature of those constraints that we elicit.

In particular, we are concerned with ensuring that this sequence behaves appropriately with respect to s and f (i.e. the declarative aspects of goals) and in particular that the *persistence*, *unachieved* and *possible* properties are reflected this sequence. The precise relationship between these states is discussed in more detail in Section 3 where we give the operational semantics for goals.

We denote by $exec(\mathsf{Goal}(s,P,f),S)$ a sequence of states that results from executing $\mathsf{Goal}(s,P,f)$ in the state S. Note that there can be more than one such sequence (as P may contain a number of possible courses of action). However, in such cases an implementation will have to select a particular course of action to follow, and the properties discussed in the remainder of this section are independent of this choice.

Thus the *persistent*, *unachieved* and *possible* properties are captured as the following properties of the sequence of states:

- If there is a state S_n in $\mathcal{S}$ such that $S_n \models s \vee f$, then $\mathcal{S} = \langle S_0, S_1, \ldots S_n \rangle$ (i.e. $\mathcal{S}$ is finite and S_n is the final state in the sequence) and $\forall\, 0 \leq i \leq n-1$ we have $S_i \not\models s \vee f$.
- Otherwise, $\mathcal{S}$ is infinite (and hence $\forall i \geq 0$ we have $S_i \not\models s \vee f$).

The *persistent* property is captured by the sequence being infinite unless there is a state in which s or f is true. The *unachieved* and *possible* properties are captured by the requirement that if there is such a state in the sequence, then

[5] More generally, P can be any construct of a plan-language.

it is the final such state and neither s nor f is true in any previous state. In other words, the execution of a goal stops if (and only if!) either s or f become true.

The *known* condition corresponds to maintaining a data structure containing a record of the goals held by the agent; this is done in Section 3; consistency is discussed in Section 4.

We now define *exec*. This provides the link between the sequence discussed above and the plans P used to achieve the goal. As discussed above, the desired behaviour of *exec* is that if s or f hold in S then it should do nothing, otherwise it executes P. There are now three cases: (i) $s \vee f$ becomes true during the execution of P, (ii) $s \vee f$ is true when P completes executing (but not before), and (iii) $s \vee f$ remains false while P executes, and is false when P completes. In the first case, we abort the execution of P when $s \vee f$ becomes true and return the result of partially executing P (*pexec*); in the second case we return the result of executing P. In the third case we cannot complete the execution of $\mathsf{Goal}(s,P,f)$ since neither s nor f are true; we thus continue to execute $\mathsf{Goal}(s,P,f)$ in the situation resulting from P's execution.

Hence we denote by $pexec(X,\mathsf{Goal}(s,P,f),S)$ the sequence of states resulting from executing *no further than the completion of* P, but halting as soon as any condition in the set X becomes true (formally, $pexec(X,P,S)$ is a prefix of $exec(P,S)$); thus $pexec(X,\mathsf{Goal}(s,P,f),S)$ has three possible cases:

- $\langle S = S_0, S_1, S_2, \ldots S_n \rangle$ where $S_n \models s \vee f \vee \bigvee_{x \in X} x$ and $\forall\, 0 \leq i < n$ we have $S_i \not\models s \vee f \vee \bigvee_{x \in X} x$
- $\langle S = S_0, S_1, S_2, \ldots S_m \rangle$ where $\forall\, 0 \leq i \leq m$ we have $S_i \not\models s \vee f \vee \bigvee_{x \in X} x$ and P terminates in state S_m
- $\langle S = S_0, S_1, S_2, \ldots \rangle$ where $\forall\, i \geq 0$ we have $S_i \not\models s \vee f \vee \bigvee_{x \in X} x$

Note that in the third case, P does not terminate and there is no state in which $s \vee f$ becomes true.

We then define $exec(\mathsf{Goal}(s,P,f))$ as the result of repeatedly applying $pexec(P)$ until one of s or f becomes true, as below.

Definition 1 *We say a sequence of states S is s,f-*compatible *if either*

- $\forall S_i \in S S_i \not\models s \vee f$
- $S = \langle S_1, S_\infty, \ldots S_\rangle$, $S_n \models s \vee f$ *and* $\forall 0 \leq i \leq n-1$ *we have* $S_i \not\models s \vee f$

We define $pexec(\mathsf{Goal}(s,P,f),S)$ *as a* maximal s,f-compatible *sequence of states that results from the execution of P commencing in state S.*

We define $exec(\mathsf{Goal}(s,P,f),S)$ *as*

- $\langle S \rangle$ *if* $S \models s \vee f$
- $pexec(\{s,f\},P,S)$ *if* $pexec(\{s,f\},P,S) \models s \vee f$ *or* $pexec(\{s,f\},P,S)$ *is infinite*
- $pexec(\{s,f\},P,S) \circ tl(exec(\mathsf{Goal}(s,P,f),S'))$ *otherwise, where S' is the final state of* $pexec(\{s,f\},P,S)$, *"$\circ$" denotes sequence concatenation, and $tl(-)$ is the sequence without its first element; this is needed since S' is also the first state of* $exec(\mathsf{Goal}(s,P,f),S')$.

We say that the goal $\mathsf{Goal}(s,P,f)$ succeeds *from S if* $exec(\mathsf{Goal}(s,P,f),S)$ *is finite and its final state S_n is such that $S_n \models s$. We say that the goal* $\mathsf{Goal}(s,P,f)$ fails *from S if* $exec(\mathsf{Goal}(s,P,f),S)$ *is finite and its final state S_n is such that $S_n \models f$.*

It is then straightforward to show the following result.

Proposition 1 *For any S_i in* $exec(\mathsf{Goal}(s,P,f),S)$ *we have that*

- *If $S_i \not\models s \vee f$ then there is a successor state S_{i+1} in* $exec(\mathsf{Goal}(s,P,f),S)$
- *Otherwise S_i is the final state in* $exec(\mathsf{Goal}(s,P,f),S)$

Thus the derived execution semantics for goals satisfies the *unachieved*, *possible* (via f), and *persistent* conditions. The *known* condition is addressed in Section 3 and *consistency* is discussed in Section 4.

3 Operational Semantics

We now present a detailed operational semantics for a high-level plan language incorporating as first-class citizens goals with both declarative and procedural aspects. The CAN[6] notation and its semantics clearly (and formally) illustrate how a possible plan language could be implemented. *Our reason for doing this is that we intend that the notion of goal developed be realisable in implemented agent systems.* The notation we present below is illustrative of the plan languages of typical agent languages, both

[6] CAN = Conceptual Agent Notation

in the BDI tradition, and elsewhere. In particular, it is similar to Rao's AgentSpeak(L) (Rao, 1996) and to Kinny's Ψ (Kinny, 2001), both of which attempt to extract the essence of a class of implemented BDI agent platforms.

3.1 The CAN Notation

An agent program (denoted by Π) consists of a collection of plan clauses of the form[7] $e : c_1 \leftarrow c_2 : P$ where e is an event, c_i are context conditions (logical formulae over the agent's beliefs) which must be true in order for the plan to applicable[8], and P is the plan body.

Beliefs (b) are first order terms but could be orthogonally extended to other logics. All that we assume are that we have operations to check whether a condition follows from a belief set ($B \models c$), to add a belief to a belief set ($B \cup \{b\}$), and to delete a belief from a belief set ($B \setminus \{b\}$). Traditionally, agent systems have implemented addition and deletion of beliefs as literal addition and deletion from a set of terms, however, there is no reason why belief revision could not be applied. A condition is a logical formula over belief terms:

$$C ::= b \mid C \wedge C \mid C \vee C \mid \neg C \mid \exists x.C$$

The plan body P is built up from the following constructs. We have the basic constants *true*, *fail* and $ex(X)$ (exception), and the primitives *act* (an action not further specified), operations to add ($+b$) and delete ($-b$) beliefs, a test for a condition ($?c$), and events[9] $!e$. We also have a number of compounds: sequencing ($P_1;P_2$), parallelism ($P_1 \| P_2$), and goals ($\mathsf{Goal}(s,P,f)$).

In addition to these compounds there are also a number of auxilliary compound forms which are used internally when assigning semantics to constructs: for example when an event matches a set of guarded plans these are collected into a set of guarded alternatives ($(\!|B:P,\ldots|\!)$). The other auxilliary compound form is a choice operator dual to sequencing ($P_1 \triangleright P_2$), which executes P_1 and then executes P_2 only if P_1 failed. The language is described by the following grammar:

$$\begin{array}{rcl} P & ::= & true \mid fail \mid ex(X) \\ & \mid & act \mid +b \mid -b \mid ?c \mid !e \\ & \mid & P;P \mid P \parallel P \mid \mathsf{Goal}(s,P,f) \\ & \mid & P \triangleright P \mid (\!|B:P,\ldots,B:P|\!) \end{array}$$

[7] An omitted c_i is equivalent to *true*.

[8] More precisely, c_1 is an eager condition and c_2 is lazy; both must be true for the plan to be applied, but they are tested at different times: c_1 is tested when the event is matched against the plans, whereas c_2 is tested when the plan is being considered for execution.

[9] Where it is obvious that e is an event we shall sometimes elide the exclamation mark, in the interests of readability.

3.2 Formal Semantics

We assume that we are given operations to check whether a condition follows from a belief set ($B \models c$), to add a belief to a belief set ($B \cup \{b\}$), and to delete a belief from a belief set ($B \setminus \{b\}$). In the case of beliefs being a set of ground atoms these operations are respectively consequence checking, and set addition/deletion.

We define a basic configuration $S = \langle B, \mathcal{G}, P \rangle$ where B is the beliefs of the agent, $\mathcal{G}$ is a set of goals being pursued (used for consistency checking – see section 4), and P is the plan being executed. A transition $S_0 \longrightarrow S_1$ specifies that executing S_0 a single step yields S_1. We define $S_0 \longrightarrow^* S_n$ in the usual way: S_n is the result of one or more single step transitions. The transition relation is defined using rules of the form $\overline{S \longrightarrow S'}$ or of the form $\dfrac{S' \longrightarrow S_r}{S \longrightarrow S'_r}$; the latter are conditional with the top (numerator) being the premise and the bottom (denominator) being the conclusion. In order to make the presentation more readable we use the convention that where a component of S isn't mentioned it is the same in S and S' (and in S_r and S'_r). We also assume that B refers to the agent's beliefs, and elide angle brackets. Thus each of the left rules is shorthand for the corresponding right rule. The full set of rules are given in figure 1.

$$\frac{B \models c}{?c \longrightarrow true}\ ?c \qquad \frac{B \models c}{\langle B, \mathcal{G}, ?c \rangle \longrightarrow \langle B, \mathcal{G}, true \rangle}\ ?c$$

$$\frac{P_1 \longrightarrow P'}{P_1;P_2 \longrightarrow P';P_2}\ ; \qquad \frac{\langle B, \mathcal{G}, P_1 \rangle \longrightarrow \langle B', \mathcal{G}', P' \rangle}{\langle B, \mathcal{G}, P_1;P_2 \rangle \longrightarrow \langle B', \mathcal{G}', P';P_2 \rangle}\ ;$$

The first rule above specifies that the condition test $?c$ transitions to true if the condition c is a consequence of the agent's beliefs ($B \models c$). The second rule specifies that $P_1;P_2$ transitions to $P';P_2$ where P' is the result of a single execution step of P_1.

The operational semantics carry around a set of conditions being watched for ($\longrightarrow_X$). This is used by goals to interrupt the execution of a sub-plan should a condition become true. In order for this to work correctly we require that the *check* rule take precedence over other rules – it must be applied if it is applicable. We add rules to propagate exceptions, interrupting further processing ($;_x$, $\|_{x1}$, $\|_{x2}$, $\triangleright_x$, and G_{xi}).

The rules for executing goals are based on the definitions of *exec* and *pexec*. The rules G and G_{copy} add s and f to the conditions being watched for (X) and execute P. This relies on the *check* rule to stop executing if s or f hold, throwing an exception $ex(x)$ when this occurs. The exception is handled appropriately by one of the three G_{xi} rules. The rule G_{tf} continues attempting to achieve the goal in the case that executing P did not lead to s or f becoming true.

We extend simple configurations (which correspond to a single thread of execution within an agent) to agent configurations $S_A = \langle N, B, G, \mathbb{P}P \rangle$ which consist of a name, a single (shared) belief set, a goal set, and a set of executing plans. The following rule defines the operational semantics over agent configurations in terms of the operational semantics over simple configurations.

$$\frac{P \in \Gamma \quad \langle B, G, P \rangle \longrightarrow \langle B', G', P' \rangle}{\langle N, B, G, \Gamma \rangle \longrightarrow \langle N, B', G', (\Gamma \setminus \{P\}) \cup \{P'\} \rangle} \; Agent$$

Note that there are elements of nondeterminism in CAN, such as the choice of plan to execute from a set of (multiple) applicable plans. In addition, the *Agent* rule nondeterministically selects an executing plan. In general, this choice should be constrained by the desired scheduling policy (such as a round-robin strategy to ensure fairness of some description).

One of the fundamental features of CAN is that it is a high-level plan language, in the spirit of process algebras such as the π-calculus and agent systems such as Ψ, rather than a programming language per se. This means that we can concentrate on the important issues, such as plan selection, event handing, belief updates, etc. rather than potentially cumbersome details such as data structures and mechanisms for passing data around. As a result, CAN is agnostic with respect to such issues.

Proposition 2 *Let $S^{(\prime)}$ be approximated by $B^{(\prime)}$ (i.e. $S \models c$ iff $B \models c$) and let $Q \in \{true, fail\}$ and $R \in \{true, fail, ex(x)\}$. Then $exec(P,S) = \langle \ldots, S' \rangle$ iff $\langle B, \mathcal{G}, P \rangle \longrightarrow^* \langle B', \mathcal{G}', Q \rangle$ and $pexec(X,P,S) = \langle \ldots, S' \rangle$ iff $\langle B, \mathcal{G}, P \rangle \longrightarrow_X{}^* \langle B', \mathcal{G}', R \rangle$.*

Proposition 3 *Let $S^{(\prime)}$ be approximated by $B^{(\prime)}$. The goal G succeeds (resp. fails) from state S iff $\langle B, \mathcal{G}, G \rangle \longrightarrow^* \langle B', \mathcal{G}', true \rangle$ (resp. if $\langle B, \mathcal{G}, G \rangle \longrightarrow^* \langle B', \mathcal{G}', fail \rangle$).*

These rules specify a precise and implementable operational semantics for a plan language including goals with both declarative and procedural aspects. The semantics of goal execution respect the desired properties of goals (namely *persistent*, *unachieved*, *possible*, and *known*; *consistency* is discussed in the next section). These rules have been translated into Prolog, yielding a prototype CAN interpreter.

3.3 An Example

Let us return to the cat example given in section 1. A set of plans suitable for sating hunger are given in figure 2. These plans include explicitly human intervention, as described in section 1.

We assume the following[10]:

$$\begin{array}{rcl}
G & = & \text{Goal}(\text{atfood}, !findfood, \text{nofood}) \\
af & = & foodat(X) \wedge location(X) \\
nf & = & not(foodat(X)) \\
P & = & findfood \\
G_1 & = & \text{Goal}_P(\text{atfood}, \{\!|!survey_terrain; reachfood|\!\}, \text{nofood}) \\
rf & = & planpath(Y,X); ?path(A); followpath \\
G_2 & = & \text{Goal}_P(\text{atfood}, !survey_terrain; reachfood \triangleright \{\!|\,|\!\}, \text{nofood}) \\
G_3 & = & \text{Goal}_P(\text{atfood}, true; reachfood \triangleright \{\!|\,|\!\}, \text{nofood}) \\
G_4 & = & \text{Goal}_P(\text{atfood}, reachfood \triangleright \{\!|\,|\!\}, \text{nofood}) \\
G' & = & \text{Goal}(\text{atfood}, \text{nofood}) \\
X & = & \{\text{atfood}, \text{nofood}\} \\
B_0 & = & \{location(floor), canjump(floor, chair), \\
 & & canjump(chair, table), canjump(ledge, table), \\
 & & canjump(chair, ledge)\} \\
B_1 & = & \{location(floor), foodat(table), \\
 & & canjump(floor, chair), canjump(chair, ledge), \\
 & & canjump(chair, table), canjump(ledge, table)\}
\end{array}$$

We consider the cat seeking food, i.e. executing the goal G. Note that the execution consists of multiple derivations, each advancing the agent a single execution step. The first step unfolds the goal, adding it to the goal set and noting that *nofood* and *atfood* are conditions that need to be watched, then posts the event *findfood* which matches a single plan which first surveys the terrain then (attempts to) reach food.

$$\dfrac{\dfrac{\dfrac{\overline{B_0 \models foodat(table) \wedge location(floor)}}{B_0, \{G'\}, !findfood \longrightarrow_X B_0, \{G'\}, \{\!|t :!st; !rf|\!\}}\; Ev}{B_0, \{G'\}, G_P \longrightarrow_\emptyset B_0, \{G'\}, G_1}\; \mathsf{G}}{B_0, \emptyset, G \longrightarrow_\emptyset B_0, \{G'\}, G_1}\; G_{copy}$$

The second step selects the (only) plan.

$$\dfrac{\dfrac{\overline{B_0 \models true}}{B_0, \{G'\}, \{\!|t :!st; !rf|\!\} \longrightarrow_X B_0, \{G'\}, !st; !rf \triangleright \{\!|\,|\!\}}\; Sel}{B_0, \{G'\}, G_1 \longrightarrow_\emptyset B_0, \{G'\}, G_2}\; \mathsf{G}$$

The third step performs the first step of the plan, namely surveying the terrain[11]. We assume that this succeeds and

[10] Due to space limitations we use the following abbreviations; st: survey_terrain, rf: reachfood, nf: nofood, af: atfood, t: true, f: fail.

[11] We have compressed matching the event against a plan, selecting that plan, and executing it into a single step.

$$\frac{B \models c}{?c \longrightarrow true}\ ?c_t \qquad \frac{B \not\models c}{?c \longrightarrow fail}\ ?c_f \qquad \frac{}{B, +b \longrightarrow B \cup \{b\}, true}\ +b \qquad \frac{}{B, -b \longrightarrow B \setminus \{b\}, true}\ -b$$

$$\frac{}{a \longrightarrow true}\ act \qquad \frac{B \models x \in X}{P \longrightarrow_X ex(x)}\ check$$

$$\frac{(t_i{:}c_i \leftarrow b_i{:}P_i) \in \Pi \quad B \models c_i}{!e \longrightarrow \langle\!\langle b_1{:}P_1, \ldots, b_n{:}P_n \rangle\!\rangle}\ Ev \qquad \frac{\neg \exists b_i{:}P_i \in \Delta . B \models b_i}{\langle\!\langle \Delta \rangle\!\rangle \longrightarrow fail}\ Sel_f \qquad \frac{b_i{:}P_i \in \Delta \quad B \models b_i}{\langle\!\langle \Delta \rangle\!\rangle \longrightarrow P_i \triangleright \langle\!\langle \Delta \setminus \{b_i{:}P_i\} \rangle\!\rangle}\ Sel$$

$$\frac{P_1 \longrightarrow P'}{P_1; P_2 \longrightarrow P'; P_2}\ ; \qquad \frac{}{true; P \longrightarrow P}\ ;_t \qquad \frac{}{fail; P \longrightarrow fail}\ ;_f \qquad \frac{}{ex(X); P \longrightarrow ex(X)}\ ;_x$$

$$\frac{P_1 \longrightarrow P'}{P_1 \| P_2 \longrightarrow P' \| P_2}\ \|_1 \qquad \frac{P_2 \longrightarrow P'}{P_1 \| P_2 \longrightarrow P_1 \| P'}\ \|_2 \qquad \frac{}{true \| P \longrightarrow P}\ \|_{t1} \qquad \frac{}{P \| true \longrightarrow P}\ \|_{t2}$$

$$\frac{}{fail \| P \longrightarrow fail}\ \|_{f1} \qquad \frac{}{P \| fail \longrightarrow fail}\ \|_{f2} \qquad \frac{}{ex(X) \| P \longrightarrow ex(X)}\ \|_{x1} \qquad \frac{}{P \| ex(X) \longrightarrow ex(X)}\ \|_{x2}$$

$$\frac{P_1 \longrightarrow P'}{P_1 \triangleright P_2 \longrightarrow P' \triangleright P_2}\ \triangleright \qquad \frac{}{true \triangleright P \longrightarrow true}\ \triangleright_t \qquad \frac{}{fail \triangleright P \longrightarrow P}\ \triangleright_f \qquad \frac{}{ex(X) \triangleright P \longrightarrow ex(X)}\ \triangleright_x$$

$$\frac{G \cup \{\mathsf{Goal}(s,f)\}, \mathsf{Goal}_P(s,P,f) \longrightarrow G', P'}{G, \mathsf{Goal}(s,P,f) \longrightarrow G', P'}\ \mathsf{G}_{copy} \qquad \frac{P \longrightarrow_{X \cup \{s,f\}} P'}{\mathsf{Goal}_P(s,P,f) \longrightarrow_X \mathsf{Goal}_P(s,P',f)}\ \mathsf{G}$$

$$\frac{Q \in \{fail, true\}}{\mathsf{Goal}_P(s,Q,f) \longrightarrow \mathsf{Goal}_P(s,P,f)}\ \mathsf{G}_{tf} \qquad \frac{}{G, \mathsf{Goal}_P(s, ex(s), f) \longrightarrow G \setminus \{\mathsf{Goal}(s,f)\}, true}\ \mathsf{G}_{x1}$$

$$\frac{}{G, \mathsf{Goal}_P(s, ex(f), f) \longrightarrow G \setminus \{\mathsf{Goal}(s,f)\}, fail}\ \mathsf{G}_{x2} \qquad \frac{y \notin \{s,f\}}{G, \mathsf{Goal}_P(s, ex(y), f) \longrightarrow G \setminus \{\mathsf{Goal}(s,f)\}, ex(y)}\ \mathsf{G}_{x3}$$

$$\frac{P \in \Gamma \quad \langle B, G, P \rangle \longrightarrow \langle B', G', P' \rangle}{\langle N, B, G, \Gamma \rangle \longrightarrow \langle N, B', G', (\Gamma \setminus \{P\}) \cup \{P'\} \rangle}\ Agent$$

Note that in rules such as *act* we assume that actions always succeed. In order to take potential failures into account, it is not difficult to modify such rules to explicitly check a given success (or failure) condition. Note also that the Ev rule does not allow for multiple solutions of context conditions, modifying it to allow for multiple solutions is straightforward. Likewise, adding explicit substitutions is not hard.

Figure 1: Operational Semantics

```
human_intervention ← -canjump(chair,table) ; +canjump(chair,ledge).
becomehungry ← +hungry ; goal(not(hungry), satehunger, fail).
satehunger ← reachfood ; eatfood.
eatfood : foodat(X) ∧ location(X) ← act(eatfood) ; -foodat(X) ; sated.
sated : hungry ← -hungry.
sated ← true.
reachfood ← goal(foodat(X) ∧ location(X),findfood, not(foodat(X))).
findfood : foodat(X) ∧ location(Y) ← survey_terrain ; planpath(Y,X) ; ?path(A) ; followpath.
survey_terrain ← act(survey_terrain).
planpath(X,Y): plan(P) ← -plan(P) ; planpath(X,Y).
planpath(X,Y): canjump(X,Y) ← +path([X,Y]).
planpath(X,Y): canjump(Y,X) ← +path([X,Y]).
planpath(X,Y): canjump(X,Z) ∧ canjump(Z,Y) ← +path([X,Z,Y]).
planpath(X,Y): canjump(X,Z) ∧ canjump(Z,Q) ∧ canjump(Q,Y) ← +path([X,Z,Q,Y]).
followpath: path([P1]) ← -path([P1]).
followpath: path([P1,P2|Ps]) ← jump(P1,P2) ; -path([P1,P2|Ps]) ; +path([P2|Ps]) ; followpath.
jump(floor,chair): location(floor) ∧ canjump(floor,chair)
        ← act(jump(floor,chair)) ; -location(floor) ; +location(chair) ; human_intervention.
jump(P1,P2): location(P1) ∧ (canjump(P1,P2) ∨ canjump(P2,P1)) ← act(jump(P1,P2)) ; -location(P1) ; +location(P2).
```

Figure 2: Sample Plans for the Cat

updates the agent's beliefs from B_0 to B_1.

$$\dfrac{\dfrac{\dfrac{\vdots}{B_0,\{G'\},!st \longrightarrow_X B_1,\{G'\},t}\;;}{B_0,\{G'\},!st;!rf \longrightarrow_X B_1,\{G'\},t;!rf}}{\dfrac{B_0,\{G'\},!st;!rf \triangleright \emptyset \longrightarrow_X B_1,\{G'\},!st;!rf \triangleright \emptyset}{B_0,\{G'\},G_2 \longrightarrow_\theta B_1,\{G'\},G_3}\;\mathsf{G}}\;\triangleright$$

The fourth step simply removes the *true*.

$$\dfrac{\dfrac{}{B_1,\{G'\},t;!rf \longrightarrow_X B_1,\{G'\},!rf}\;\mathit{st}}{\dfrac{B_1,\{G'\},t;!rf \triangleright \emptyset \longrightarrow_X B_1,\{G'\},!rf \triangleright \emptyset}{B_1,\{G'\},G_3 \longrightarrow_\theta B_1,\{G'\},G_4}\;\mathsf{G}}\;\triangleright$$

The execution continues by:

- planning a path from the floor to the table (via the chair),
- jumping onto the chair
- having the human move the chair
- realising that the next step in the plan (jumping from the chair to the table) cannot be done
- surveying the terrain
- planning a path from the chair to the table (via the ledge)
- jumping to the ledge, jumping to the table, and then eating[12].

[12] In this run there isn't any food on the ledge

4 Reasoning about Goals

When an agent has more than one goal to pursue, there are a number of ways in which the pursuit of these goals can fail to be independent. At one extreme an agent has as goals both A and $\neg A$, here it is clearly irrational for the agent to attempt to satisfy both goals (or at least both goals simultaneously). It is also possible for goals to be logically consistent but not simultaneously achievable (for example they might require the same resources). A third possibility is that only some of the plans for each goal conflict, and hence it is possible to achieve both goals simultaneously by an appropriate choice of plan. For example, consider a goal to eat food (and there is food on the table and on a shelf) and a goal to scratch the table (an obviously naughty cat). These two goals can be simultaneously pursued and achieved (e.g. jumping onto the table and eating the food on it while scratching it), but some choice of plan (e.g. planning to eat the food on the shelf) will create a conflict between the goals[13]. In some instances it may also be appropriate to show that pursuit of a given set of goals can be achieved independently of each other — in other words, that there is no interference at all, and hence the plans to achieve each goal can be freely interleaved. Finally, we may wish to determine instances of positive interference, i.e. situations in which the achievement of one goal can

[13] Of course, it is possible that all plans to achieve the goals conflict. We are currently working on solutions to such instances by identifying such conflicts via summary information based on the work of Clement and Durfee (Clement and Durfee, 1999b; Clement and Durfee, 1999a).

assist in the achievement of other goals.

In this section we discuss various possibilities for performing these kinds of reasoning in our framework. For space reasons, this will necessarily be illustrative of the possibilities rather than a detailed prescription.

Our framework for goals simplifies the development of this kind of reasoning, such as being able to identify sub-goals $sg^n(G)$ which *necessarily* appear in the pursuit of G and sub-goals $sg^p(G)$ which *possibly* so appear. The definition for $sg^n(G)$ is thus

$$\begin{aligned} sg^n(P_1;P_2) &= sg^n(P_1) \cup sg^n(P_2) \\ sg^n(\{B_1 : P_1, \ldots, B_n : P_n\}) &= \bigcap_{1 \le i \le n} sg^n(P_i) \\ sg^n(\mathsf{Goal}(s,P,f)) &= \{\mathsf{Goal}(s,P,f)\} \cup sg^n(P) \\ sg^n(a) &= \emptyset \end{aligned}$$

and similarly for $sg^p(G)$ with $\bigcap$ replaced with $\bigcup$. Note that $sg^n(G) \subseteq sg^p(G)$.

In general an agent will have a set of goals $\mathcal{G}$ and it wants to ensure that the addition of a given goal G does not conflict with $\mathcal{G}$. We say that G and $\mathcal{G} = \{G_1, \ldots, G_n\}$ are *necessarily consistent* iff all possible subgoals of G and all possible subgoals of all G_i in $\mathcal{G}$ do not conflict, i.e. $\forall g \in sg^p(G).\forall G_i \in \mathcal{G}.\forall g_i \in sg^p(G_i).g \natural g_i$, where $G_1 \natural G_2$ indicates that the s and f conditions for G_1 and G_2 are compatible[14]. Similarly, we say that G and $\mathcal{G} = \{G_1, \ldots, G_n\}$ are *necessarily inconsistent* iff some necessary subgoal of G and some necessary subgoal of any G_i in $\mathcal{G}$ conflict, i.e. $\exists g \in sg^n(G).\exists G_i \in \mathcal{G}.\exists g_i \in sg^n(G_i).\neg(g \natural g_i)$.

The definition of necessarily (in)consistent only considers *goal compatibility* in detecting conflict between a new goal G and the existing set of goals $\mathcal{G}$. However, we also need to consider the resource requirements of goals and ensure that there are no conflicts with respect to the resource requirements of G and that of $\mathcal{G}$.

We can define notions of necessary and possible resources as we did for sub-goals, but there are some differences in the way we deal with goal compatibility and *resource compatibility*. In goal compatibility we compare each sub-goal of G with every sub-goal of every goal G_n in $\mathcal{G}$ and this is sufficient. However this would not suffice for resource compatibility. For example assume that there are 50 units of the resource *energy* available, G requires 20 units of *energy*, and G_n has two *necessary* sub-goals SG_1 and SG_2 that require 25 units of energy each. G is compatible with SG_1 and SG_2 individually (since $20+25 \le 50$) but not with G_n because G_n requires the combined resources of SG_1 and SG_2 (i.e. $20+(25+25) \not\le 50$). Therefore it is necessary to combine the resource requirements of the sub-goals of a goal in order to check for resource compatibility.

[14] For example, that achieving one goal does not cause the other goal to fail.

In order to combine resource summaries we need to identify the different types of resources. Similar to sub-goals resources can also be classified as either *necessary resources* or *possible resources*. Some resources are no longer available after use (*consumable resources*, e.g. food), whilst others are still available after use (*reusable resources*, e.g. chair). The manner in which we derive resources summaries for a goal depends on the classification of the resource and the way in which the sub-goals are combined (i.e. whether they are sequential sub-goals or parallel subgoals). Due to space limitation we cannot provide formal definitions and algorithms for deriving and using resource requirement summaries for goals, however we have addressed them in other work (Thangarajah et al., 2002b) which we will discuss in section 5.

If G and $\mathcal{G}$ are neither necessarily consistent nor necessarily inconsistent then they are *possibly consistent* (or, for that matter, *possibly inconsistent*). In the case where two goals are necessarily consistent we can achieve them concurrently. If they are necessarily inconsistent then we need to choose between them (Thangarajah et al., 2002a). *If they are possibly inconsistent then we need to choose consistent means of achieving these goals.*

Our framework also provides a suitable foundation for reasoning about positively interacting goals. We say that two goals G_1 and G_2 *necessarily support* each other if there is common necessary subgoal, i.e. $sg^n(G_1) \cap sg^n(G_2) \neq \emptyset$. For example, driving to the beach and driving to the garage might both have the necessary subgoal of getting more petrol ("Gas" in American English). However, this requires that any plans chosen must generate the *same* single sub-goal, which appears to be too strong, where it seems more natural to require that for any plan choice, a common sub-goal, possibly depending on the plan choice made, can be made to arise. We thus define a weaker condition and say that two goals G_1 and G_2 *possibly support* each other ($G_1 \rightleftharpoons G_2$) if they either necessarily support each other or if (i) given a choice of P_i from the set of plans $\{P_1, \ldots, P_n\}$ associated with G_1 there exists a choice of Q_j from the set of plans $\{Q_1, \ldots, Q_m\}$ associated with G_2, such that $P_i \rightleftharpoons Q_j$; and (ii) given a choice of Q_j there exists a choice of P_i such that $P_i \rightleftharpoons Q_j$.

Note that it is entirely possible for both positive and negative interactions to occur simultaneously. Consider the following example (based on (Horty and Pollack, 2001)): we need to travel to the airport to catch a 4pm flight and are considering catching either a taxi (expensive but faster) or a bus (cheaper but slower). We need to decide whether to adopt the goal of attending a meeting at 2:30pm. Assuming the meeting is likely to run late and is being held at the university this would constrain us to catch a taxi and thus attending the meeting is seen as less desirable *in con-*

text. If the meeting were to be held at 3:30 at the university then it would prevent us from reaching the airport in time, which makes attending the meeting impossible (assuming the flight is more important than the meeting). On the other hand, if the meeting were held at the airport, then, since we are already intending to travel to the airport, the cost of attending the meeting is *lower* than it would be, and hence in this context, the meeting is seen as possible and desirable. This example illustrates the sorts of reasoning that we want to be able to perform, namely assessing how goals can both hinder and help other goals.

5 Discussion

We have proposed an explicit goal construct for agent systems, based on the desired (declarative) properties of goals, and we have given an operational interpretation, via the rules of Section 3, which realises these properties. We have also discussed how reasoning about interactions between goals, both positive and negative, can be performed within our goal framework. We anticipate that this work will form the basis of significant development of agent systems with explicit representation of goals, including those based on the popular BDI model, thus reducing the gap between theory and practice, and enhancing the intelligent capabilities of such systems.

Due to space limitations we did not provide details on how to derive resource summaries and the details of how they can be used to detect conflict between goals with respect to resource requirements. We have however addressed this in (Thangarajah et al., 2002b). In that work we characterise different types of resources and define resource requirements summaries. We give algorithms for deriving resource requirements, using resource requirements to detect conflict, and performing dynamic updates of resource requirements; we also discuss how conflict can be resolved.

Although the operational behaviour of goals can be realised in BDI systems (using maintenance conditions), we believe that goals are a sufficiently key concept for intelligent agents that they deserve to be represented directly. Furthermore, the declarative aspects are important and by representing goals we enable reasoning about interactions (both positive and negative) to be done.

5.1 Related Work

AgentSpeak (Rao, 1996) and Ψ (Kinny, 2001) both capture in a formal way the semantics of a plan language interpreter. They focus on capturing the essence of current practice and as a result capture the weaknesses of (current) BDI systems including the lack of declarative goals and associated problems.

The GOAL language (Hindriks et al., 2001) uses declarative goals but lacks a sufficiently powerful notion of procedural goals: plans cannot use sequences and are limited to being reactive.

Horty and Pollack (Horty and Pollack, 2001) formalise the reasoning process for positively interacting (i.e. assisting) goals by defining a notion of compatible plans and of the merging of (compatible) plans. The cost of a plan $\mathcal{P}$ *in context* C (where the two are compatible) is the difference between the cost of the context merged with the plan and the cost of the context on its own: $\kappa(\mathcal{P}/C) = \kappa(\mathcal{P} \cup C) - \kappa(C)$. Their work only addresses positive interactions and assumes that plans can be (at least roughly) simulated and that the cost of executing a plan can be at least approximated[15]. By comparison, we reason directly about conflicts between goals and address both negative and positive interactions.

There is a considerable amount of work on conflict in agent systems which addresses many different types of conflicts at various levels (see for example (Tessier et al., 2000)). By contrast, the main contribution of this paper isn't ways of dealing with conflict, but rather a realisation of goals in BDI-like systems. This realisation provides an implementable *foundation* upon which conflict resolution can be built, and alternative approaches compared.

5.2 Future Work

There is a wide range of directions for future work including: reasoning about plans; looking at attaching declarative information to program elements other than goals (in particular to plans); investigating the use of other forms of declarative assertions in addition to postconditions (e.g. preconditions, in-conditions (Clement and Durfee, 1999b; Clement and Durfee, 1999a)); investigating how Goal(s, P, f) allows for the derivation of potential plans P' from s (and f) should the given plan P fail in achieving s; extending to richer forms of goals including temporal logic and resources; and investigating the use of declarative aspects of goals in debugging agent systems.

We have addressed the issue of dealing with resource requirements in detecting conflicts between goals. Future work includes extending this concept to handle positive interaction (co-operation) between goals with respect to resource requirements (e.g. there is a positive interaction between two goals if one goal renews a consumable resource that is necessary for the other goal).

[15] For practical reasons they do not compute the precise cost, but rather maintain an approximation interval which contains the true cost. In some cases decisions can be made based on the interval without having to know the precise cost.

Acknowledgements

We would like to acknowledge the support of Agent Oriented Software Pty. Ltd. and of the Australian Research Council (ARC) under grant CO0106934.

References

Busetta, P., Rönnquist, R., Hodgson, A., and Lucas, A. (1998). JACK Intelligent Agents - Components for Intelligent Agents in Java. Technical report, Agent Oriented Software Pty. Ltd, Melbourne, Australia.

Clement, B. J. and Durfee, E. H. (1999a). Identifying and resolving conflicts among agents with hierarchical plans. In *AAAI Workshop on Negotiation: Settling Conflicts and Identifying Opportunities, Technical Report WS-99-12*. Available from *http://ai.eecs.umich.edu/people/bradc/*.

Clement, B. J. and Durfee, E. H. (1999b). Theory for coordinating concurrent hierarchical planning agents using summary information. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 495–502. Available from *http://ai.eecs.umich.edu/people/bradc/*. A longer technical report version is also available.

d'Inverno, M., Kinny, D., Luck, M., and Wooldridge, M. (1998). A formal specification of dMARS. In Singh, M., Rao, A., and Wooldridge, M., editors, *Intelligent Agents IV: Proceedings of the Fourth International Workshop on Agent Theories, Architectures, and Languages*, pages 155–176. Springer-Verlag LNAI 1365.

Giacomo, G. D., Lespérance, Y., and Levesque, H. (2000). ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121:109–169.

Hindriks, K., de Boer, F., van der Hoek, W., and Meyer, J.-J. (2001). Agent programming with declarative goals. In *Intelligent Agents VI - Proceedings of the 7th International Workshop on Agent Theories, Architectures, and Languages (ATAL'2000)*. Springer Verlag.

Hindriks, K. V., Boer, F. S. D., der Hoek, W. V., and Meyer, J.-J. C. (1999). Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401.

Horty, J. and Pollack, M. (2001). Evaluating new options in the context of existing plans. *Artificial Intelligence*, 127:199–220.

Huber, M. J. (1999). JAM: A BDI-theoretic mobile agent architecture. In *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 236–243.

Ingrand, F. F., Georgeff, M. P., and Rao, A. S. (1992). An architecture for real-time reasoning and system control. *IEEE Expert*, 7(6).

Jennings, N. and Wooldridge, M. (1998). Applications of intelligent agents. In Jennings, N. R. and Wooldridge, M. J., editors, *Agent Technology: Foundations, Applications, and Markets*, chapter 1, pages 3–28. Springer.

Jennings, N. R. (2001). An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41.

Kinny, D. (2001). The psi calculus: an algebraic agent language. In *Agent Theories, Architectures, and Languages (ATAL-2001)*.

Padgham, L. and Lambrix, P. (2000). Agent capabilities: Extending BDI theory. In *Proceedings of Seventeenth National Conference on Artificial Intelligence - AAAI 2000*, pages 68–73.

Rao, A. and Georgeff, M. (1998). Decision procedures for BDI logics. *Journal of Logic and Computation*, 8(3):292–342.

Rao, A. S. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. In de Velde, W. V. and Perrame, J., editors, *Agents Breaking Away: Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96)*, pages 42–55. Springer Verlag. LNAI, Volume 1038.

Rao, A. S. and Georgeff, M. P. (1991). Modeling rational agents within a BDI-Architecture. In Allen, J., Fikes, R., and Sandewall, E., editors, *Principles of Knowledge Representation and Reasoning, Proceedings of the Second International Conference*, pages 473–484.

Rao, A. S. and Georgeff, M. P. (1992). An abstract architecture for rational agents. In Rich, C., Swartout, W., and Nebel, B., editors, *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 439–449, San Mateo, CA. Morgan Kaufmann Publishers.

Tessier, C., Chaudron, L., and Müller, H.-J., editors (2000). *Conflicting Agents: Conflict Management in Multi-Agent Systems*. Kluwer Academic Publishers. ISBN 0-7923-7210-7.

Thangarajah, J., Padgham, L., and Harland, J. (2002a). Representation and reasoning for goals in BDI agents. In *Australasian Computer Science Conference*.

Thangarajah, J., Winikoff, M., Padgham, L., and Fischer, K. (2002b). Avoiding resource conflicts in BDI agents. Submitted to the European Conference on Artificail Intelligence (ECAI 2002).

van Linder, B., van der Hoek, W., and Meyer, J.-J. (1995). Formalising motivational attitudes of agents: On preferences, goals and commitments. In Wooldridge, M., Müller, J., and Tambe, M., editors, *Intelligent Agents II: Agent Theories, Architectures, and Languages, IJCAI'95 Workshop (ATAL)*, pages 17–32. Springer, LNAI 1037.

Winikoff, M., Padgham, L., and Harland, J. (2001). Simplifying the development of intelligent agents. In Stumptner, M., Corbett, D., and Brooks, M., editors, *AI2001: Advances in Artificial Intelligence. 14th Australian Joint Conference on Artificial Intelligence*, pages 555–568. Springer, LNAI 2256.

Wooldridge, M. (1998). Agent-based computing. *Interoperable Communication Networks*, 1(1):71–97.

Adapting Golog for Composition of Semantic Web Services

Sheila McIlraith
Knowledge Systems Laboratory
Department of Computer Science
Stanford University
Stanford, CA 94305-9020
sam@ksl.stanford.edu

Tran Cao Son
Computer Science Department
New Mexico State University
PO Box 30001, MSC CS
Las Cruces, NM 88003, USA
tson@cs.nmsu.edu

Abstract

Motivated by the problem of automatically composing network accessible services, such as those on the World Wide Web, this paper proposes an approach to building agent technology based on the notion of generic procedures and customizing user constraint. We argue that an augmented version of the logic programming language Golog provides a natural formalism for automatically composing services on the Semantic Web. To this end, we adapt and extend the Golog language to enable programs that are generic, customizable and usable in the context of the Web. Further, we propose logical criteria for these generic procedures that define when they are *knowledge self-sufficient* and *physically self-sufficient*. To support information gathering combined with search, we propose a middle-ground Golog interpreter that operates under an assumption of reasonable persistence of certain information. These contributions are realized in our augmentation of a ConGolog interpreter that combines online execution of information-providing Web services with offline simulation of world-altering Web services, to determine a sequence of Web Services for subsequent execution. Our implemented system is currently interacting with services on the Web.

1 INTRODUCTION

Two important trends are emerging in the World Wide Web (WWW). The first is the proliferation of so-called *Web Services* – self-contained, Web-accessible programs and devices. Familiar examples of Web services include information-gathering services such as the map service at yahoo.com, and world-altering services such as the book-buying service at amazon.com. The second WWW trend is the emergence of the so-called *Semantic Web*. In contrast to today's Web, which is designed primarily for human interpretation and use, the Semantic Web is a vision for a future Web that is unambiguously computer-interpretable [2]. This will be realized by marking up Web content, its properties, and its relations, in a reasonably expressive markup language with a well-defined semantics. DAML+OIL, a description-logic based Semantic Web markup language, is one such language [10, 11].

Our interest is in the confluence of Web Services and the Semantic Web. In early work, we outlined a semantic markup for describing the capabilities of Web services, initially in first-order logic and a predecessor to DAML+OIL [19]. This effort evolved into a coalition of researchers from BBN, CMU, Nokia, SRI, Stanford and Yale who are developing a DAML+OIL ontology for Web services, called DAML-S [4]. Several metaphors have proved useful in developing this markup, including viewing Web services as functions with inputs and outputs, and alternatively as primitive and complex actions with (knowledge) preconditions and (knowledge) effects. While DAML-S is not yet complete, two versions of the ontology have been released for public scrutiny [3]. We will return to DAML-S towards the end of the paper.

The provision of, effectively, a knowledge representation of the properties and capabilities of Web services enables the automation of many tasks, as outlined in [19]. In this paper we focus on the task of automated Web service composition (WSC): Given a set of Web services and a description of some task or goal to be achieved (e.g., "Make the travel arrangements for my KR2002 conference trip."), find a composition of services that achieves the task. Disregarding network issues, WSC can be conceived as either a software synthesis problem, or as a planning and plan execution problem, depending upon how we represent our services. In either case, this application domain has many distinctive features that require and support tailoring. We identify and address many of these features in this paper.

In this paper we conceive WSC as a planning and execution task, where the actions (services) may be complex actions. In related work, we show how to compile service representations into operators that embody all the possible evolutions of a complex action, in order to treat complex actions as primitive action plan operators [18]. As a planning task, WSC is distinguished in that it is planning with very incomplete information. Several sequenced information-gathering services may be required, that culminate in the execution of only a few world-altering services. (Imagine making your travel plans on the Web.) Since our actions (services) are software programs, the input and output parameters of the program act as knowledge preconditions and knowledge effects in a planning context. Software programs can also have side-effects in the world (such as the purchase of a commodity), that are modeled as non-knowledge effects. Service preconditions are regularly limited to knowledge preconditions. Information-gathering services (aka sensors) don't fail, network issues aside. Exogenous events affect the things being sensed. Persistence of knowledge has a temporal extent associated with it (contrast stock prices to the price of a shirt at the Gap), which affects the sequencing of services. Services often provide multiple outputs, a subset of which must be selected to act as input for a subsequent service (consider picking flights).

Many services perform similar functions, so WSC must choose between several services, each sharing some of the same effects. Also, plans (compositions of services) are often short, so the plan search space is short and broad. WSC tasks may or may not be described in terms of a goal state. In some instances they are described as a set of loosely coupled goals, or constraints. Many plans may satisfy the WSC task. User input and user constraints are key in pruning the space of plans (e.g., choosing from the multitude of available flights) and in distinguishing desirable plans.

The unique features of WSC serve to drive the work presented in this paper. Rather than realizing WSC simply as planning, we argue that a number of the activities a user may wish to perform on the (semantic) WWW or within some networked service environment, can be viewed as customizations of reusable, high-level generic procedures. For example, we all use approximately the same generic procedure to make our travel plans, and this procedure is easily described. Nevertheless, it is difficult to task another person, less a computer, to make your travel plans for you. The problem lies not in the complexity of the procedure, but rather in selecting services and options that meet your individual constraints and preferences. Our vision is to construct reusable, high-level generic procedures, and to archive them in sharable (DAML-S) generic-procedures ontologies so that multiple users can access them. A user could then select a task-specific generic procedure from the ontology and submit it to their agent for execution. The agent would automatically customize the procedure with respect to the user's personal or group-inherited constraints, the current state of the world, and available services, to generate and execute a sequence of requests to Web services to perform the task.

We realize this vision by adapting and extending the logic programming language Golog (e.g., [14, 22, 6]). The adaptations and extensions described in the sections to follow are designed to address the following desiderata of our WSC task. **Generic:** We want to build a class of programs that are sufficiently generic to meet the needs of a variety of different users. Thus programs will often have a high degree of nondeterminism to embody the variability desired by different users. **Customizable:** We want our programs to be easily customizable by individual users. **Usable:** We want our programs to be usable by different agents with different a priori knowledge. As a consequence, we need to ensure that the program accesses all the knowledge it needs, or that certain knowledge is stipulated as a prerequisite to executing the program. Similarly, the program ensures the actions it might use are $Poss$-ible. The programs must be both knowledge and physically self-sufficient.

2 ADAPTING GOLOG

Golog (e.g., [14, 8, 6, 22]) is a high-level logic programming language, developed at the University of Toronto, for the specification and execution of complex actions in dynamical domains. It is built on top of the situation calculus (e.g.,[22]), a first-order logical language for reasoning about action and change. Golog was originally developed to operate without considering sensing (aka information-gathering) actions. For Web applications, we rely on a version of Golog built on top of the situation calculus with knowledge and sensing actions (e.g., [24, 22]), which we henceforth refer to simply as the situation calculus.

In the situation calculus [6, 22], the state of the world is described by functions and relations (fluents) relativized to a situation s, e.g., $f(\vec{x}, s)$. To deal with sensing actions, a special knowledge fluent K, whose first argument is also a situation, is introduced. Informally, $K(s', s)$ holds if the agent is in the situation s but believes (s)he might be in s'. The function $do(a, s)$ maps a situation s and an action a into a new situation. A situation s is simply a history of the primitive actions performed from an initial, distinguished situation S_0. A situation calculus theory $\mathcal{D}$ comprises the following sets of axioms (See [22] for details.):

- domain-independent foundational axioms of the situation calculus, Σ;
- accessibility axioms for K, $\mathcal{K}_{init}$[1];

[1] At this stage, we do not impose any conditions on K such as

- successor state axioms, $\mathcal{D}_{SS}$, one for K and one for every domain fluent F;
- action precondition axioms, $\mathcal{D}_{ap}$, one for every action a in the domain, that serve to define $Poss(a, s)$;
- axioms describing the initial situation, $\mathcal{D}_{S_0}$ (including axioms about K);
- unique names axioms for actions, $\mathcal{D}_{una}$;
- domain closure axioms for actions, $\mathcal{D}_{dca}$[2].

Golog builds on top of the situation calculus by providing a set of extralogical constructs for assembling primitive actions, defined in the situation calculus, into complex actions that collectively comprise a program, δ. Constructs include the following.

a — primitive actions
$\delta_1; \delta_2$ — sequences
$\phi?$ — tests
$\delta_1 \mid \delta_2$ — nondeterministic choice of actions
$(\pi x)\delta(x)$ — nondeterministic choice of arguments
δ^* — nondeterministic iteration
if ϕ **then** δ_1 **else** δ_2 **endIf** – conditionals
while ϕ **do** δ **endWhile** — while loops

Note that the conditional and while-loop constructs are actually defined in terms of other constructs.

if ϕ **then** δ_1 **else** δ_2 **endIf** $\doteq [\phi?; \delta_1] \mid [\neg\phi?; \delta_2]$
while ϕ **do** δ **endWhile** $\doteq [\phi?; \delta]^*; \neg\phi?$

These constructs can be used to write programs in the language of a domain theory, e.g.,
$buyAirTicket(\vec{x})$;
if far **then** $rentCar(\vec{y})$ **else** $bookTaxi(\vec{y})$ **endIf**.

Given a domain theory, $\mathcal{D}$ and Golog program δ, program execution must find a sequence of actions $\vec{a}$ such that: $\mathcal{D} \models Do(\delta, S_0, do(\vec{a}, S_0))$. $Do(\delta, S_0, do(\vec{a}, S_0))$ denotes that the Golog program δ, starting execution in S_0 will legally terminate in situation $do(\vec{a}, S_0)$, where $do(\vec{a}, S_0)$ abbreviates $do(a_n, do(a_{n-1}, \ldots, do(a_1, S_0)))$.

2.1 CUSTOMIZING GOLOG PROGRAMS

In this section we extend Golog to enable individuals to customize a Golog program by specifying personal constraints. To this end, we introduce a new distinguished fluent in the situation calculus called $Desirable(a, s)$, i.e., action a is desirable in situation s. We contrast this with $Poss(a, s)$, i.e. action a is physically possible in situation s. We further restrict the cases in which an action is executable by requiring not only that an action a is $Poss(a, s)$ but further that it is $Desirable(a, s)$. This further constrains the search space for actions when realizing a Golog program. The set of *Desirable* fluents, one for each action, is referred to as $\mathcal{D}_D$. $Desirable(a, s) \equiv true$ unless otherwise noted.

An individual specifies her personal constraints in our Semantic Web markup language. The constraints are expressed in the situation calculus as *necessary conditions for an action a to be desirable*, $\mathcal{D}_{necD}$ of the form:

$$Desirable(a, s) \supset \omega_i, \tag{1}$$

and *personal constraints*, $\mathcal{D}_{PC}$ which are situation calculus formulae, C.

For example, Marielle would like to buy an airline ticket from origin o to destination d, if the driving time between these two locations is greater than 3 hours. Thus $Desirable(buyAirTicket(o, d, dt), s) \supset gt(DriveTime(o, d), 3, s)$ is included in $\mathcal{D}_{necD}$. Similarly, Marielle has specified dates she must be at home and her constraint is not to be away on those dates. Thus, $\mathcal{D}_{PC}$ includes: $\neg(Away(dt, s) \wedge MustbeHome(dt, s))$. Using $\mathcal{D}_{necD}$ and $\mathcal{D}_{PC}$, and exploiting our successor state axioms and domain closure axioms for actions, $\mathcal{D}_{SS}$ and $\mathcal{D}_{dca}$, we define $Desirable(a, s)$ for every action a as follows:

$$Desirable(A(\vec{x}), s) \equiv \Omega_A \wedge \bigwedge_{C \in \mathcal{D}_{PC}} \Omega_{PC}, \tag{2}$$

where $\Omega_A = \omega_1 \vee \ldots \vee \omega_n$, for each ω_i of (1). E.g.,
$\Omega_{buyAirTicket} = gt(DriveTime(o, d), 3, s)$, and

$$\Omega_{PC} \equiv \mathcal{R}[C(do(A(\vec{x}), s))]$$

where $\mathcal{R}$ is repeated regression rewriting (e.g., [22]) of $C(do(A(\vec{x}), s))$, the constraints relativized to $do(A(\vec{x}), s)$, using the successor state axioms, $\mathcal{D}_{SS}$ from $\mathcal{D}$. E.g.,

$$\Omega_{PC} \equiv \mathcal{R}[\neg(Away(dt, do(buyAirTicket(o, d, dt), s)) \wedge MustbeHome(dt, do(buyAirTicket(o, d, dt), s)))]$$

We rewrite this expression using the successor state axioms for fluents $Away(dt, s)$ and $MustbeHome(dt, s)$. E.g.,

$$Away(dt, do(a, s)) \equiv [(a = buyAirTicket(o, d, dt) \wedge d \neq Home) \vee (Away(dt, s) \wedge \neg(a = buyAirTicket(o, d, dt) \wedge d = Home))]$$

$$MustbeHome(dt, do(a, s)) \equiv MustbeHome(dt, s)$$

From this we determine:

$$Desirable(buyAirTicket(o, d, dt), s) \equiv gt(DriveTime(o, d), 3, s) \wedge (d = Home \vee \neg MustbeHome(dt, s)) \tag{3}$$

Having computed $\mathcal{D}_D$, we include it in $\mathcal{D}$[3]. In addition to computing $\mathcal{D}_D$, the set of *Desirable* fluents, we also modify the computational semantics of our dialect of Golog.

that K to be reflexive, symmetric, transitive or Euclidean in the initial situation. (See [22, pg. 302-308].)

[2]Not always necessary, but we will require it in 2.1.

[3]Henceforth, all reference to $\mathcal{D}$ includes $\mathcal{D}_D$.

In particular, we adopt the *computational semantics* for Golog. (See [6] for details.) Two predicates are used to define the semantics. $Trans(\delta, s, \delta', s')$ is intended to say that the program δ in situation s may legally execute one step, ending in situation s' with the program δ' remaining. $Final(\delta, s)$ is intended to say that the program δ may legally terminate in situation s. We require one change in the definition to incorporate *Desirable*. In particular, (4) is replaced by (5).

$$Trans(a, s, \delta', s') \equiv Poss(a, s) \wedge \delta' = nil \wedge s' = do(a, s) \quad (4)$$

$$Trans(a, s, \delta', s') \equiv Poss(a, s) \wedge Desirable(a, s) \wedge \delta' = nil \wedge s' = do(a, s) \quad (5)$$

We can encode this more compactly by simply defining $Legal(a, s) \equiv Poss(a, s) \wedge Desirable(a, s)$, and replacing *Poss* with *Legal* in (4). This approach has many advantages. First it is elaboration tolerant [17]. An individual's customized $\mathcal{D}_D$ may simply be added to an existing situation calculus axiomatization. If an individual's constraints change, the affected *Desirable* fluents in $\mathcal{D}_D$ may be elaborated by a simple local rewrite. Further, *Desirable* is easily implemented as an augmentation of most existing Golog interpreters. Finally, it reduces the search space for terminating situations, rather than pruning situations after they have been found. Thus, it has computational advantages over other approaches to determining preferred sequences of actions. Our approach is related to the approach to the qualification problem proposed by Lin and Reiter [15]. There are other types of customizing constraints which we do not address in this paper (e.g., soft and certain temporal constraints). We address these constraints in future work.

2.2 ADDING THE ORDER CONSTRUCT

In the previous subsection we described a way to customize Golog programs by incorporating user constraints. In order for Golog programs to be customizable and generic, they must have some nondeterminism to enable a variety of different choice points to incorporate user's constraints. Golog's nondeterministic choice of actions construct (|) and nondeterministic choice of arguments construct (π) both provide for nondeterminism in Golog programs.

In contrast, the sequence construct (;) provides no such flexibility, and can be overly constraining. Consider the program: $buyAirTicket(\vec{x})$; $rentCar(\vec{y})$. The ";" construct dictates that $rentCar(\vec{y})$ must be performed in the situation resulting from performing $buyAirTicket(\vec{x})$ and that $Poss(rentCar(\vec{y}), do(buyAirTicket(\vec{x}), s))$ must be true, otherwise the program will fail. Imagine that the precondition $Poss(rentCar(\vec{y}), s)$ dictates that the user's credit card not be over its limit. If *Poss* is not true, we would like for the agent executing the program to have the flexibility to perform a sequence of actions to reduce the credit card balance, in order to achieve this precondition, rather than having the program fail. The sequence construct ";" does not provide for this flexibility.

To enable the insertion of actions in between a sequence of actions, for the purposes of achieving preconditions, we define a new construct called *order*, designated by the ":" connective[4]. Informally, $a_1 : a_2$ will perform the sequence of action $a_1; a_2$ whenever $Poss(a_2, do(a_1, s))$ is true. However, when it is false, the ":" construct dictates that Golog search for a sequence of actions $\vec{a}$ that achieves $Poss(a_2, do(\vec{a}, do(a_1, s)))$. This can be achieved by a planner that searches for a sequence of actions $\vec{a}$ to achieve the goal $Poss(a_2, do(\vec{a}, do(a_1, s)))$. To simplify this paper, we restrict a_2 to be a primitive action. The definition is easily extended to an order of complex actions $\delta_1 : \delta_2$. Thus, $a_1 : a_2$ is defined as:

a_1; **while** $(\neg Poss(a_2))$ **do** $(\pi a)[Poss(a)?; a]$ **endWhile**; a_2

It is easy to see that the while loop will eventually achieve the precondition for a_2 if it can be achieved.

We extend the computational semantics to include ":".

$$Trans(\delta : a, s, \delta', s') \equiv Trans((\delta; achieve(Poss(a)); a, s, \delta', s') \quad (6)$$

$$Final(\delta : a, s) \equiv Final(\delta; achieve(Poss(a)); a, s) \quad (7)$$

where $achieve(G)$ = **while** $(\neg G)$ **do** $(\pi a)[Poss(a)?; a]$ **endWhile**. Since *achieve* is defined in terms of existing Golog constructs, the definitions of *Trans* and *Final* follow from previous definitions.

Note that ":" introduces undirected search into the instantiation process of Golog programs and though well-motivated for many programs, should be used with some discretion because of the potential computational overhead. We can improve upon this simplistic specification by a more directed realization of the action selection mechanism used by *achieve* using various planning algorithms.

Also note that the order construct has been presented here independently of the notion of *Desirable*, introduced in the previous subsection. It is easy to incorporate the contributions of Section 2.2 by replacing $achieve(Poss(a))$ with $achieve(Legal(a))$ in Axioms (6) and (7) above, plus any other deontic notions we may wish to include. Finally note that a variant of the order construct also has utility in expressing narrative as proposed in [21]. We can modify ":" to express that actions a_1 and a_2 are ordered, but that it is not necessarily the case that a_2 occurred in situation $do(a_1, s)$.

[4] Created from a combination of existing constructs.

2.3 SELF-SUFFICIENT PROGRAMS

Now that our Golog programs are customizable and can be encoded generically, we wish them to be usable. Sensing actions are used when the agent has incomplete knowledge of the initial state (often true for WSC), or when exogenous actions exist that change the world in ways the agent's theory of the world does not predict. Web service compositions often have the characteristic of sequences of information-gathering services, performed to distinguish subsequent world-altering services. In our work, we need to define Golog programs that can be used by a variety of different agents without making assumptions about what the agent knows. As such, we want to ensure that our Golog programs are self-sufficient with respect to obtaining the knowledge that they require to execute the program. Further, we wish our programs to ensure that all preconditions for actions the program tries to execute are realized within the program, or are established as an explicit precondition of the program.

To make this concrete, we define the notion of a Golog program δ being *self-sufficient* with respect to an action theory $\mathcal{D}$ and *kernel initial state*, $Init_\delta$. $Init_\delta$ is a formula relativized to (suppressed) situation s, denoting the necessary preconditions for executing δ. To characterize self-sufficiency, we introduce the predicate $ssf(\delta, s)$. $ssf(\delta, s)$ is defined inductively over the structure of δ.

$$ssf(nil, s) \equiv true \tag{8}$$

$$ssf(\phi?, s) \equiv \mathbf{KWhether}^5(\phi, s) \tag{9}$$

$$ssf(a, s) \equiv \mathbf{KWhether}(Poss(a, s)) \wedge \mathbf{KWhether}(Desirable(a, s)) \tag{10}$$

$$ssf(\delta_1; \delta_2, s) \equiv ssf(\delta_1, s) \wedge \forall s'.[Do(\delta_1, s, s') \supset ssf(\delta_2, s')] \tag{11}$$

$$ssf(\delta_1|\delta_2, s) \equiv ssf(\delta_1, s) \wedge ssf(\delta_2, s) \tag{12}$$

$$ssf(\delta^*, s) \equiv ssf(\delta, s) \wedge \forall s'.[Do(\delta, s, s') \wedge ssf(\delta^*, s')] \tag{13}$$

$$ssf((\pi x)\delta(x), s) \equiv \forall x.ssf(\delta(x), s) \tag{14}$$

$$ssf(\textbf{if } \phi \textbf{ then } \delta_1 \textbf{ else } \delta_2 \textbf{ endIf}, s) \equiv \mathbf{KWhether}(\phi, s) \wedge (\phi(s) \supset ssf(\delta_1, s)) \wedge (\neg\phi(s) \supset ssf(\delta_2, s)) \tag{15}$$

$$ssf(\textbf{while } \phi \textbf{ do } \delta \textbf{ endWhile}, s) \equiv \mathbf{KWhether}(\phi, s) \wedge (\phi(s) \supset ssf(\delta, s) \wedge (\forall s'.[Do(\delta, s, s') \supset ssf(\textbf{while } \phi \textbf{ do } \delta \textbf{ endWhile}, s')])) \tag{16}$$

Since ":" is defined in terms of existing constructs, $ssf(\delta_1 : \delta_2, s)$ follows from (8)–(14) above.

[5] **KWhether**(ϕ, s) abbreviates a formula indicating that the truth value of ϕ is known in situation s [24].

Definition 1 (KSSF: Knowledge Self-Sufficient Program) *$KSSF(\delta, Init_\delta)$, Golog program δ is knowledge self-sufficient relative to action theory $\mathcal{D}$ and kernel initial state $Init_\delta$ iff $\mathcal{D} \models Init_\delta(S_0)$ and $\mathcal{D} \models ssf(\delta, S_0) \equiv true$.*

$KSSF(\delta, Init_\delta)$ ensures that given $Init_\delta$, execution of the Golog program δ will not fail for lack of knowledge. However, the program may fail because it may be im-$Poss$-ible to perform an action. $KSSF$ ensures that the agent knows whether $Poss$ is true, but not that it actually *is* true. To further ensure that our generic procedures are physically self-sufficient, we define $PSSF(\delta, Init_\delta)$.

Definition 2 (PSSF: Physically Self-Sufficient Program) *$PSSF(\delta, Init_\delta)$, Golog program δ is physically self-sufficient relative to action theory $\mathcal{D}$ and kernel initial state $Init_\delta$ iff $KSSF(\delta, Init_\delta)$ and $\mathcal{D} \models \exists s'.Do(\delta, S_0, s')$.*

Proposition 1 *For every Golog program δ and associate kernel initial state $Init_\delta$, $PSSF(\delta, Init_\delta) \supset KSSF(\delta, Init_\delta)$.*

This follows directly from Definitions 1-2.

Next we discuss how to verify KSSF and PSSF for a common subset of Golog programs.

We call a Golog program, δ *loop-free* if it does not contain the nondeterministic iteration and while-loop constructs. Note that we may preserve the loop-free nature of our programs, while using these programming constructs by defining a maximum iteration count, or time-out. It follows that.

Proposition 2 *For every loop-free Golog program $(\delta, Init_\delta)$ and associated situation calculus theory $\mathcal{D}$, there exist first-order situation calculus formulae ϕ_P and ϕ_K such that*

$$KSSF(\delta, Init_\delta) \equiv \phi_K \text{ and } PSSF(\delta, Init_\delta) \equiv \phi_P,$$

and ϕ_P and ϕ_K do not mention ssf.

The proof is inductive over the structure of δ.

From these propositions, it follows that $PSSF$ and $KSSF$ of loop-free programs, $(\delta, Init_\delta)$ can be verified using regression followed by theorem proving in the initial situation. For programs with potentially unlimited looping, $\phi_K(s)$ and $\phi_P(s)$ are not first-order definable, and hence are problematic.

Proposition 3 *Let δ be a loop-free Golog program, and let ϕ_K and ϕ_P be defined as in Proposition 2. Let $\mathcal{K}_{init}$ consist of any subset of the accessibility relations Reflexive, Symmetric, Transitive, Euclidean, then*

1. $KSSF(\delta, Init_\delta)$ iff $D_{una} \cup D_{S_0} \cup \mathcal{K}_{init} \models \mathcal{R}[\phi_K]$

2. $PSSF(\delta, Init_\delta)$ iff $D_{una} \cup D_{S_0} \cup \mathcal{K}_{init} \models \mathcal{R}[\phi_P]$.

This follows directly from Reiter's regression theorem with knowledge [22].

We wish to highlight the work of Ernie Davis on Knowledge Preconditions for Plans [5], which we became aware of when we first presented *ssf* [20]. There are many similarities to our work. One significant difference is that he makes no distinction between (what we distinguish as) knowledge sufficiency and physically sufficiency in his framework, i.e., for a plan to be executable, he requires that the agent has the knowledge to execute it and that it must be physically possible. Further, we construct the *ssf* condition from the situation calculus theory for primitive actions that can be regressed over situations and verified in the initial situation. He develops a set of rules that can be used to check for plan executability. The set of rules, is sufficient but not necessary.

3 EXECUTING GOLOG PROGRAMS

Now that we have defined customizable, generic and usable generic procedures for WSC, we must execute them. In building a Golog interpreter that incorporates sensing actions, the interplay between sensing and execution of world-altering actions can be complex and a number of different approaches have been discussed (e.g., [7, 13, 22]). While [7] and [22] advocate the use of an online interpreter to reason with sensing actions, [13] suggests the use of an offline interpreter with conditional plans. The trade-off is clear. An online interpreter is incomplete because no backtracking is allowed, while an offline interpreter is computationally expensive due to the much larger search space, and the need to generate conditional plans, if sensing actions are involved. The choice between an online and offline interpreter depends on properties of the domain, and in particular, since exogenous actions can affect the value of fluents, on the temporal extent of the persistence of the information being sensed. In a robotics domain, an online interpreter is often more appropriate, whereas an offline interpreter is more appropriate for contingency planning.

3.1 MIDDLE-GROUND EXECUTION

We define a middle ground between offline and online execution, which we argue is appropriate for a large class of Semantic Web WSC applications. Our middle-ground interpreter (MG) senses online to collect the relevant information needed in the Golog program, while only simulating the effects of world-altering actions. By executing sensing actions rather than branching and creating a conditional plan, MG reduces search space size, while maintaining the ability to backtrack by merely simulating world-altering actions, initially. The outcome is a sequence of world-altering actions that are subsequently executed[6]. Humans often follow this approach, collecting information on the Web (e.g., flight schedules) while only simulating the world-altering actions (buying tickets, etc.) in their head until they have a completed plan to execute.

Of course, the veracity of MG is predicated on an important assumption – that the information being gathered, and upon which world-altering actions are being selected, persists. We assume that the fluents MG is sensing persist for a reasonable period of time, and that none of the actions in the program cause this assumption to be violated. This assumption is generally true of much of the information we access on the Web (e.g., flight schedules, store merchandise), but not all (e.g., stock prices). This assumption is much less pervasive in mobile robotic applications where we may assume persistence for milliseconds, rather than minutes or hours. We formalize this assumption as follows.

Definition 3 (Conditioned-on Fluent) *Fluent C is a conditioned-on fluent in Golog program δ iff δ contains the Golog construct $\phi?$ and C appears in formula ϕ.*

Recall that the $\phi?$ construct is used to define the conditional (if-then-else) and the while-loop constructs. It is also commonly used within the δ of $(\pi x)\delta(x)$.

Definition 4 (Invocation and Reasonable Persistence (IRP) Assumption) *Golog program and kernel initial state $(\delta, Init_\delta)$ adhere to the invocation and reasonable persistence assumption if*

1. Non-knowledge preconditions for sensing actions are true in $\mathcal{D}_{S_0} \cup Init_\delta(S_0)$.

2. Knowledge of preconditions for actions and conditioned-on fluents C in δ, once established, persists[7].

Condition 1 ensures that all sensing actions can be executed by the MG interpreter. Condition 2 ensures that decisions are predicated on correct information. Condition 1 may seem extreme, but, as we argued earlier in this paper, by their nature, Web services generally only have knowledge preconditions. The persistence of knowledge in Condition 2, trivially holds from the frame assumption for knowledge. This condition addresses change by subsequent or exogenous actions.

We claim that under the IRP assumption, MG does the right thing for programs that are physically self-sufficient.

Claim 1 (Veracity of MG) *Given an action theory D and*

[6] At this stage they can alternately be shown to a human for approval before execution. Our interpreter can also generate and present multiple alternate courses of action.

[7] I.e., no subsequent actions inside or outside the program change the value of sensed fluents.

Golog program δ such that $PSSF(\delta, Init_\delta)$, and $(\delta, Init_\delta)$ adheres to IRP, let $\vec{\alpha}_w$ be the sequence of world-altering actions selected by the middle-ground interpreter, MG, for subsequent execution. Assuming no exogenous actions and no sensor errors[8]*, it follows that executing $\vec{\alpha}_w$ yields the same truth value for all fluents F in $\mathcal{D}$ as an online interpreter with an oracle that chooses $\vec{\alpha}_w$ at the appropriate branch points in its interpretation of δ.*

Let $do(\vec{\alpha}, S_0)$ be the terminating situation, following execution of Golog program δ with theory $\mathcal{D}$, using the interpreter MG, starting in S_0. Then $\mathcal{D} \models Do(\delta, S_0, do(\vec{\alpha}, S_0))$, and we denote the sequence of actions $\vec{\alpha}$ by the relation $\mathrm{MG}(\mathcal{D}, \delta, Init_\delta, \vec{\alpha})$.

$\vec{\alpha}$ is comprised of both sensing actions and world-altering actions (e.g., $[s_1, a_1, a_2, a_3, s_2, a_4]$). Let $\vec{\alpha}_s$ be the sequence of sensing actions in $\vec{\alpha}$ (i.e., $[s_1, s_2]$), and likewise let $\vec{\alpha}_w$ be the sequence of world-altering actions in $\vec{\alpha}$ (i.e., $[a_1, a_2, a_3, a_4]$). MG executes the sensing actions $\vec{\alpha}_s$, interleaved with the simulation of world-altering actions, searching to find the appropriate terminating situation, $\vec{\alpha}$. MG then outputs $\vec{\alpha}$. $\vec{\alpha}_w$, the subsequence of world-altering actions, are extracted from $\vec{\alpha}$ and executed in the world.

Theorem 1 *Given an action theory D and Golog program δ such that $PSSF(\delta, Init_\delta)$, and $(\delta, Init_\delta)$ adheres to IRP, suppose $MG(\mathcal{D}, \delta, Init_\delta, \vec{\alpha})$ holds for some $\vec{\alpha}$. Assume that there are no sensor errors, and that no exogenous actions affect fluents in $\mathcal{D}$, then for all fluents F in $\mathcal{D}$*

$$\mathcal{D} \models F(\vec{x}, do(\vec{\alpha}_w, do(\vec{\alpha}_s), S_0)) \equiv F(\vec{x}, do(\vec{\alpha}, S_0)).$$

In cases where the IRP Assumption is at risk of being violated, the full sequence of sensing and world-altering actions generated by MG, $\vec{\alpha}$, could be re-executed with an online execution monitoring system. The system would reperform sensing actions to verify that critical persistence assumptions were not violated. In the case where the IRP Assumption does not hold for some or all conditioned-on fluents in a Golog program, MG could be integrated with an interpreter that builds conditional plans for branch points that do not adhere to IRP, following the approach proposed in [13]. The explicit encoding of search areas in a program, as proposed by [7] through the addition of their Σ search construct, can achieve some of the same functionality as our middle-ground interpreter. Indeed, the principle defined above, together with an annotation of the temporal extent of conditioned-on fluents within the action theory provides a means of automatically generating programs with embedded search operators Σ, as proposed in [7]. We leave a formal account of this to a future paper.

[8] Trivially true of virtually all current-day information-gathering Web services.

3.2 MIDDLE-GROUND PROLOG INTERPRETER

In Section 2, we proposed extensions to Golog to enable programs to be generic, customizable and self-sufficient. In Section 3, we proposed a strategy for middle-ground execution that enables an efficient and thorough combination of sensing and search. We have modified the ConGolog offline interpreter in [7, 6] to realize these enhancements. We describe the necessary code modifications in the subsections to follow, and prove the correctness of our implementation. We adopt notational conventions for Prolog code, that differ from those used for theories of action. To avoid confusion, all Prolog code is listed in courier font. Prolog variables are uppercase, and constants are lowercase, contradicting the situation calculus notational convention.

3.2.1 User customizing constraints

Personal constraints were added to the ConGolog interpreter by the following straightforward and elegant modification to the code, that accounts for the addition of the *Desirable* predicate. We replaced the following code:

```
trans(A,S,R,S1) :-  primAct(A),
  (poss(A,S), R=nil, S1=do(A,S)); fail.
```

of the ConGolog interpreter with

```
trans(A,S,R,S1) :- primAct(A),
  (poss(A,S), desirable(A,S),
     R=nil, S1=do(A,S));  fail.
```

This ensures that every action selected by the interpreter is also a desirable one.

In Section 3.3, we will show how to encode the $Desirable(a, s)$ predicate in our Prolog action theory. The domain-independent Prolog rules for the MG interpreter include the following rule to ensure that actions are desirable unless proved otherwise.

```
desirable(A,S):- \+ not_desirable(A,S).
```

3.2.2 Order Construct

To include the order construct ":", we added the following rules to our interpreter:

```
final(P:A, S):-
    action(A),
    final([P,achieve(poss(A),0),A],S).
trans(P:A,S,R,S1):-
    action(A),
    trans([P,achieve(poss(A),0),A],S,R,S1).
```

where achieve(Goal,0) is an A*-planner, adapted from the so-called *World Simplest Breath First Planner* (wsbfp) developed by Reiter [22, pg. 234]. We appeal to its simplicity and the soundness and completeness of the A* algorithm. Obviously any planner can be used to accomplish this task. We are investigating the effectiveness of other planners.

3.2.3 Sensing Actions

We incorporate sensing actions and their effects into our interpreter, using an approach that eliminates the need for the situation calculus K fluent. Recently, Soutchanski [25] proposed a somewhat similar transformation, articulating conditions under which his representation was correct. This is also similar in spirit to the approach in [8].

To accommodate both backtracking and sensing, we assume that the truth value of a certain fluent, say $F(\vec{x}, s)$, can be determined by executing *an external function call*, A. The call is denoted by $exec(A(\vec{x}), s)$. Whenever the execution succeeds, F is true; otherwise, it is false. Note that because Prolog answers queries with free variables by returning possible values for these variables, this technique is equally suitable for sensed functional fluents. This is illustrated in Example 2. The use of external function calls, together with the IRP Assumption, allows us to write equations of the following form, which are embodied into the successor state axiom of a fluent $F(\vec{x})$:

$$F(\vec{x}, do(A(\vec{x}), s)) \equiv exec(A(\vec{x}), s) \quad (17)$$

Equation (17) is translated into Prolog as follows

```
holds(f(X), do(a(X), S)):- exec(a(X), S).
```

In addition, we need to provide the set of rules that call the action a externally.

```
exec(a(X), S):- <external function call>
```

These rules are domain dependent and may be unique to the specific Prolog encoded situation calculus domain theory. In the following section, we discuss how to translate our situation calculus theories into Prolog, and illustrate how we make external function calls for WSC.

3.3 TRANSLATING SITCALC TO PROLOG

To translate a specific situation calculus theory, $\mathcal{D}$ into a set of Prolog rules, $\mathcal{D}^P$, we follow the description provided in [22], with the following additions, predominantly to accommodate sensing actions and the *Desirable* predicate.

- For each propositional fluent $F(\vec{x}, s)$, create a corresponding propositional fluent `f(X,S)` in D^P.
- For each functional fluent $F(\vec{x}, s) = \vec{y}$, create a corresponding propositional fluent `f(X, Y, S)` in D^P.
- Successor state axioms for non-sensed fluents are translated into Prolog following the description in [22]. For fluents whose truth value can change as the result of a sensing action, the normal successor state axiom encoding is augmented to include the suitable external function calls. Further, we add the necessary code to realize those calls.
- For each action $A(\vec{x})$ for which $Desirable(A(\vec{x}), s) \equiv \Omega(A(\vec{x}), s)$ is defined following Equation (2), create a Prolog rule `not_desirable(a(X),S):- \+ omega(a(X),S)`. We use Prolog's negation as failure to infer that by default an action is desirable in a situation, as per the MG code in Section 3.2.1.

We illustrate the points above with the following examples.

Example 1: We return to our simple travel theme. Many of the major airlines offer WWW programs that allow a user to buy an air ticket online. We conceive these services as world-altering actions in the situation calculus, e.g., $buyAirTicket(o, d, dt)$. To simplify the example, we disregard all parameters except origin, destination, and date (o, d, dt, respectively). This service has a number of effects including asserting that the user will own a ticket after its execution. The successor state axiom for $ownAirTicket(o, d, dt, s)$ is as follows.

$$\begin{aligned} &ownAirTicket(o, d, dt, do(a, s)) \equiv \\ &\quad (a = buyAirTicket(o, d, dt) \wedge ticketAvail(o, d, dt)) \vee \\ &\quad (a \neq buyAirTicket(o, d, dt) \\ &\qquad \wedge\ ownAirTicket(o, d, dt, s)) \end{aligned}$$

which is encoded in Prolog as follow.

```
holds(ownAirTicket(O, D, DT, do(E,S)):-
    E = buyAirTicket(O, D, DT),
    holds(ticketAvail(O, D, DT), S);
    \+ E = buyAirTicket(O, D, DT),
    holds(ownAirTicket(O, D, DT), S).
```

Recall that MG does not execute world-altering actions. Hence there is no external function to execute `buyAirTicket`. To create an interpreter that executed (some) world-altering actions immediately, the Prolog code would be modified analogously to the sensing-action code below.

Example 2: Now consider the map service offered at www.yahoo.com. Simplified, this service takes as input the origin of a trip o, and the destination d, and returns, among other things, the driving time between o and d. In the context of WSC, we view this service as the sensing action, $getDrivingTime(o, d)$ which, after its execution, tells us the value of the functional fluent $DriveTime(o, d, s)$. In Prolog, we create a corresponding propositional fluent, `driveTime(O,D,T,S)`. For simplicity, we assume $DriveTime(o, d, s)$ is only sensed by $getDrivingTime(o, d)$, but the extension to multiple actions is trivial.

The successor state axiom for `driveTime(O,D,T)` is as follows. Note the call to execute `ex_get_driving_time(O, D, T)`.

```
holds(driveTime(O, D, T), do(E,S)):-
    E = getDrivingTime(O, D),
```

```
        exec(ex_get_driving_time(O, D, T)) ;
    \+ E = getDrivingTime(O, D),
    holds(driveTime(O, D, T), S).
```

To specify how the action `ex_get_driving_time(O, D, T)` is executed, we need additional code. In our implementation, our sensing actions are all calls to Web services. We do this via a call to the Open Agent Architecture (OAA) agent brokering system [16]. OAA in turn requests a service named `get_directions`.

We first write a general rule for the generic exec call.

```
exec(A):- A.
```

We also write a rule for the actual call.

```
ex_get_driving_time(O, D, T):-
  oaa_Solve(
    get_directions(O, ' ', D, ' ', 0, X),[]),
  drvTime(X, T).
```

`oaa_Solve(get_directions(...),[])` requests the yahoo service that provides the driving time from `O` to `D`. As the information provided by the service (`X`) contains additional information including images and driving directions, we have written some additional code to extract what we need from what the service returns. The code extraction is performed by `drvTime(X, T)`. We omit the details in the interest of space. As we will discuss in the following section, when our vision of semantic Web services is realized, such extra code will be unnecessary.

Example 3: In Section 2.1, we discussed Marielle's personal preferences and showed how they translated into Equation (3). Here we show how we represent this axiom in our Prolog encoding of the domain theory.

```
not_desirable(buyAirTicket(O, D, DT), S):-
    holds(driveTime(O, D, T), S),
    T <= 3 ;
    \+ D = ''home'', holds(mustBeHome(DT, S)).
```

3.4 CORRECTNESS OF THE INTERPRETER

We complete Section 3 with a theorem that proves the correctness of our interpreter.

Theorem 2 *Given an action theory $\mathcal{D}$ and Golog program δ such that $PSSF(\delta, Init_\delta)$, and $(\delta, Init_\delta)$ adheres to IRP, if $D^P \cup MG^P \vdash Do(\delta, S_0, S)$ then there exists a model $\mathcal{M}$ of $\mathcal{D}$ such that $\mathcal{M} \models Do(\delta, S_0, S)$, where D^P is the set of Prolog rules representing $\mathcal{D}$, MG^P is the set of Prolog rules representing our MG Golog interpreter, and $\vdash$ is proof by our Prolog interpreter.*

Following [22], the action theories in this paper are definitional theories when $\mathcal{D}_{S_0}$ is complete, i.e., when $\mathcal{D}_{S_0}$ contains a definition for each fluent in the theory. This can be achieved by making the closed-world assumption (CWA). Since our programs are self-sufficient, this seems less egregious. Proposition 4 follows immediately from the Implementation Theorem of [22].

Proposition 4 *Given an action theory $\mathcal{D}$ and Golog program δ such that $PSSF(\delta, Init_\delta)$, and $(\delta, Init_\delta)$ adheres to IRP. Then, for all situations S,*

$$D^P_{CWA} \cup MG^P \vdash Do(\delta, S_0, S) \text{ iff}$$
$$\mathcal{D} \cup CWA(S_0) \models Do(\delta, S_0, S),$$

where $CWA(S_0)$ is the closed-world assumption on S_0, defined as $\{F(S_0) \equiv false \mid$ there exists no definition of F in $\mathcal{D}_{S_0}\}$, MG^P is the set of Prolog rules for our MG Golog interpreter, D^P_{CWA} is the set of Prolog rules representing $\mathcal{D} \cup CWS(S_0)$, and $\vdash$ is proof by our Prolog interpreter.

4 COMPOSING WEB SERVICES

A significant aspect of our contribution is that the research described to this point is implemented and has been tested on a running system that interacts with services on the Web. In this section, we step back and situate the agent technology we've been describing in the context of our system architecture for *Semantic Web Service Composition*. We also discuss further details of our implementation. Finally, we conclude this section with an example generic procedure that illustrates the use of our work.

4.1 ARCHITECTURE

Figure 1 illustrates the key components of our semantic WSC architecture [19]. Of course, the Semantic Web does not exist yet – www.yahoo.com does not use semantic markup such as DAML+OIL to describe its services nor to disseminate information. We describe both the architecture for our system, and in the section to follow discuss how we've accommodated for the pieces of the architecture that are not yet realizable in an elegant way.

The key features of this architecture follow.

Semantic Markup of Web Services: Individual Web services are described in a semantic Web markup language. The programs, their control structure and data flow, are described using a declarative process modeling language. Processes are either atomic or composite. Each process has inputs, outputs, preconditions and effects. It also has a grounding that describes the communication-level properties of the service. A service profile is created for describing and locating the service. Collectively, this semantic markup provides a declarative API for the service so that programs/agents can read this markup and understand how to interact with a service.

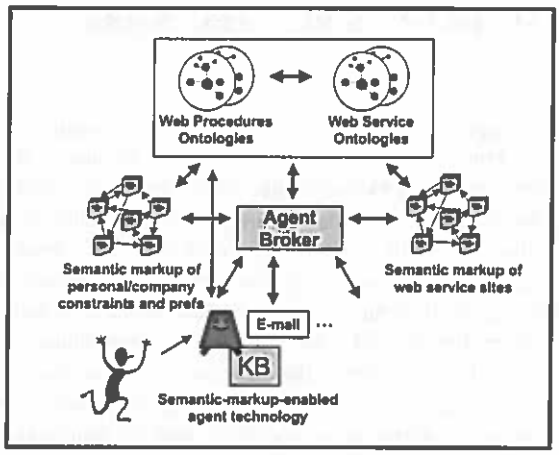


Figure 1: Semantic WSC Architecture

Ontologies of Web Services: To encourage reuse of vocabulary, parsimony, and shared semantic understanding, individual Web service descriptions are organized into Web service ontologies. For example, an ontology might contain a service class called `buy`, with subclasses `buyTicket`, `buyBook`, etc. `buyTicket` might in turn be subclassed into `buyAirTicket`, etc. Thus, a Web service provider wishing to describe a new service can simply subclass an existing service, inheriting vocabulary, and ensuring some degree of integration with existing systems.

Semantic Markup and Ontologies of Generic Procedures: Generic procedures can be described using the same semantic markup constructs used to describe Web services. After all, they are just programs. Similarly, generic procedures are stored in ontologies to facilitate sharing and reuse. The ability to share generic procedures is what motivated our desire to make procedures knowledge and physically self-sufficient.

Semantic Markup of Personal/Company Constraints: In addition to semantic markup of services, people can archive their personal preferences and constraints as semantic markup. These profiles can likewise be stored in ontologies, so that users can inherit constraints from family, their place of work, or other affiliations.

Semantic-Markup-Enabled Agent Technology: The architecture also includes a variety of agent technologies that communicate with Web services through an Agent Broker. Our Golog interpreter is one such agent technology.

Agent Broker: The agent broker accepts requests for services from the agent technology or other services, selects an appropriate service and directs the request to that service. Likewise it relays responses back to the requester.

4.2 IMPLEMENTATION

To realize our agent technology, we started with a simple implementation of an offline ConGolog interpreter in Quintus Prolog 3.2. We have modified and extended this interpreter as described in Section 3. Agent brokering is performed using the Open Agent Architecture (OAA) agent brokering system [16]. We have modified our Golog interpreter to communicate with Web services via OAA. Unfortunately, commercial Web services currently do not utilize semantic markup. In order to provide a computer-interpretable API, and computer-interpretable output, we use an information extraction program, World Wide Web Wrapper Factory[9] (W4). This program extracts the information we need from the HTML output of Web services. All information-gathering actions are performed this way. For obvious practical (and financial!) reasons, we are not actually executing world-altering services.

All the core infrastructure is working and our Golog interpreter is communicating with services on the Web via OAA. We first demoed our Golog-OAA-WWW system in August, 2000 [19]. Since then, we have been refining it and working on Semantic Web connections. Where our architecture has not been fully realized is with respect to full automation of semantic markup. We originally constructed rudimentary service and procedure ontologies in first-order logic. We are migrating these to DAML-S, as we complete our DAML-S specification. Eventually our Golog interpreter, will populate its KB from the DAML-S ontologies and from DAML+OIL ontologies of user's customizing constraints. These declarative representations will be compiled into a situation calculus representation. We have constructed partial compilers for DAML-S to PDDL[10], and for PDDL to the situation calculus, but we are still predominantly hand-coding situation calculus theories at this time.

4.3 EXAMPLE

We complete this section with an example generic procedure. Consider the example composition task given at the beginning of this paper, "Make the travel arrangements for my KR2002 conference trip." If you were to perform this task yourself using services available on the Web, you might first find the KR2002 conference Web page and determine the location and dates of the conference. Based on the location, you would decide upon the most appropriate mode of transportation. If traveling by air, you might then check flight schedules with one or more Web services, book flights, and arrange transportation to the airport through another Web service. Otherwise, you might book a rental car. You would then need to arrange transportation and accom-

[9] db.cis.upenn.edu/W4/

[10] Plan Domain Definition Language.

modations at the conference location, and so on.

We have created a generic procedure for arranging travel that captures many aspects of this example. Our generic procedure selects and books transportation (car/air), hotel, local transportation, emails the customer an itinerary, and updates an online expense claim. As noted previously, these generic procedures are not that complex – they are indeed generic. It is the interplay with user constraints that makes our approach powerful.

In what follows we provide Prolog code for a subset of our generic travel procedure. We have simplified the program slightly (particularly the number of parameters) for illustration purposes. We have also used informative constant and term names to avoid explanation. `D1` and `D2` are the departure and return dates of our trip. `pi` is the nondeterministic choice of action arguments construct, π. Sensing actions, such as `searchForRFlight()` have associated execution code, not included here. Recall that to interpret this generic procedure, Golog will look for actions that are `desirable` as well as possible.

The following is a generic procedure for booking a return airline ticket.

```
proc(bookRAirTicket(O, D, D1, D2),
[
  poss(searchForRFlight(O, D, D1,D2)) ?,
  searchForRFlight(O, D, D1, D2),
  [ pi(price,
        [ rflight(ID, price) ?,
          (price < usermaxprice) ?,
          buyRAirTicket(ID, price) ])
  ]
]).
```

Note the choice of flight based on price using π. Procedures for booking a car or hotel can be written in a similar fashion. We compose such procedures to make a Golog travel program.

```
proc(travel(D1, D2, O, D),
[
   [  bookRAirticket(O, D, D1, D2),
      bookCar(D, D, D1, D2)
   ] |
   bookCar(O, O, D1, D2),
   bookHotel(D, D1, D2),
   sendEmail,
   updateExpenseClaim
]).
```

Note the use of nondeterministic choice of actions. If booking a return air ticket or booking a car at the destination prove undesirable, Golog tries to book a car at the origin so the user can drive to the destination and back.

We have tested our generic travel procedure with different tasks and a different user constraints. These tests have confirmed the ease and versatility of our approach to WSC.

5 SUMMARY & RELATED WORK

In this paper we addressed the problem of automated Web service composition and execution for the Semantic Web. We developed and extended theoretical research in reasoning about action and cognitive robotics, implemented it and experimented with it. We addressed the WSC problem through the provision of high-level generic procedures and customizing constraints. We proposed Golog as a natural formalism for this task. As an alternative to planning, our approach does not change the computational complexity of the task of generating a composition. Nevertheless, most Web service compositions are short, and the search space is broad. Consequently, our approach has the potential to drastically reduce the search space, making it computationally advantageous. Additionally, it is compelling, and easy for the average Web user to use and customize.

Our goal was to develop Golog generic procedures that were easy to use, generic, customizable, and that were usable by a variety of users under varying conditions. We augmented Golog with the ability to include customizing user constraints. We also added a new programming construct called *order* that relaxes the notion of *sequence*, enabling the insertion of actions to achieve the precondition for the next action to be performed by the program. This construct facilitates customization as well as enabling more generic procedures. Finally, we defined the notion of knowledge and physically self-sufficient programs that are executable with minimal assumptions about the agent's initial state of knowledge, or the state of the world. We showed that these criteria could be verified using regression and theorem proving. Adherence to these criteria makes our generic procedures amenable to wide-spread use. To execute our programs, we defined a middle-ground approach to execution that performed online execution of necessary information-gathering Web services with offline simulation of world-altering services. Thus, our MG interpreter determined a sequence of world-altering Web Services for subsequent execution. We proved that our approach to execution had the intended consequences, under the IRP assumption.

These contributions were implemented as modifications to an existing ConGolog interpreter and we proved the correctness of our implementation. Further they have been intergrated into a Semantic Web Architecture, that includes an agent broker for communication with Web Services, and a variety of service-related ontologies. We have tested our results with a generic procedure for travel and a variety of different customizing constraints that showcase the effectiveness of our approach. Though our work was focused on Web service composition, the work presented in this paper has broad relevance to a variety of cognitive robotic tasks.

Much related work was identified in the body of this paper, with the work in [7, 22], and more recently, [23], being most closely related. Several other agent technologies deserve mention. The topic of agents on the internet has been popular over the years. Some of the first and most related work is the softbot work done at the University of Washington [9]. They also use action schemas to describe information-providing and world-altering actions that an agent can use to plan to achieve a goal on the internet. More recently, [26, 1, 12] have all developed some sort of agent technology that interacts with the Web.

ACKNOWLEDGEMENTS

We thank the Cognitive Robotics Group at the University of Toronto for providing an initial ConGolog interpreter that we have extended and augmented, and SRI for the use of the Open Agent Architecture software. We would also like to thank Honglei Zeng for his work on the OAA interface to our Golog code [19]. Finally, we gratefully acknowledge the financial support of the US Defense Advanced Research Projects Agency DAML Program grant number F30602-00-2-0579-P00001. The second author would also like to acknowledge the support of NSF grant NSF-EIA-981072 and NASA grant NCC2-1232.

References

[1] V. Benjamins et al. IBROW3: An Intelligent Brokering Service for Knowledge-Component Reuse on the World Wide Web. In *KAW'98, Banff, Canada*

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. In *Scientific American*, May 2001.

[3] DAML-S. http://www.daml.org/services, 2001.

[4] DAML-S Coalition: A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng. DAML-S: Semantic markup for Web services. In *Proc. Int. Semantic Web Working Symposium (SWWS)*, 411–430, 2001.

[5] E. Davis. Knowledge Preconditions for Plans. *Journal of Logic and Computation*, 4(5):721–766, 1994.

[6] G. De Giacomo, Y. Lespérance, and H. Levesque. *ConGolog*, a concurrent programming language based on the situation calculus. *AIJ*, 121(1-2):109–169, 2000.

[7] G. De Giacomo and H. Levesque. An incremental interpreter for high-level programs with sensing. In *Logical Foundations for Cognitive Agents, Contributions in Honor of Ray Reiter*, pages 86–102, 1999.

[8] G. De Giacomo and H. Levesque. Projection using regression and sensors. In *IJCAI'99*, 160–165, 1999.

[9] O. Etzioni and D. Weld. A softbot-based interface to the internet. *JACM*, pages 72–76, July 1994.

[10] J. Hendler and D. McGuinness. The DARPA agent markup language. In *IEEE Intelligent Systems Trends and Controversies*, November/December 2000.

[11] I. Horrocks, F. van Harmelen, P. Patel-Schneider, T. Berners-Lee, D. Brickley, D. Connolly, M. Dean, S. Decker, D. Fensel, P. Hayes, J. Heflin, J. Hendler, O. Lassila, D. McGuinness, and L. Stein. DAML+OIL, March 2001. http://www.daml.org/2001/03/daml+oil-index.

[12] C. Knoblock, et. al. Mixed-initiative, multi-source information assistants. In *Proceedings of the 10th International Conference on the World Wide Web*, pages 697–707, 2001.

[13] G. Lakemeyer. On sensing and off-line interpreting in Golog. In *Logical Foundations for Cognitive Agents, Contr. in Honor of Ray Reiter*, pages 173–187, 1999.

[14] H. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1-3):59–84, April-June 1997.

[15] F. Lin and R. Reiter. State constraints revisited. *Journal of Logic and Computation*, 4(5):655–678, 1994. Special Issue on Action and Processes.

[16] D. L. Martin, A. J. Cheyer, and D. B. Moran. The open agent architecture: A framework for building distributed software systems. *Applied Artificial Intelligence*, 13:91–128, January-March 1999.

[17] J. McCarthy. Mathematical logic in artificial intelligence. *Daedalus*, pages 297–311, Winter, 1988.

[18] S. McIlraith and R. Fadel. Planning with complex actions. Submitted for publication, 2002.

[19] S. McIlraith, T. Son, and H. Zeng. Semantic Web services. *IEEE Intelligent Systems (Special Issue on the Semantic Web)*, 16(2):46–53, March/April 2001.

[20] S. McIlraith and T. C. Son. Adapting ConGolog for Programming the Semantics Web. In *Working Notes of The Fifth International Symposium on Logical Formalization of Commonsense Reasoning*, pages 195–202, 2001.

[21] R. Reiter. Narratives as programs. In *Proc. of the Seventh International Conference on Knowledge Representation and Reasoning (KR2000)*, pages 99–108, 2000.

[22] R. Reiter. *KNOWLEDGE IN ACTION: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, 2001.

[23] S. Sardiña. Local conditional high-level robot programs. In *Proceedings of the 4th Workshop on Nonmonotonic Reasoning and Action, IJCAI, August 2001*, pages 195–202, 2001.

[24] R. Scherl and H. Levesque. The frame problem and knowledge producing actions. In *Proceedings of the 12th National Conference on Artificial Intelligence*, pages 689–695, 1993.

[25] M. Soutchanski. A Correspondence Between Two Different Solutions to the Projection Task with Sensing. In *Working Notes of Common Sense 2001*, pages 235–242, 2001.

[26] R. Waldinger. Deductive composition of Web software agents. In *Proc. NASA Wkshp on Formal Approaches to Agent-Based Systems, LNCS*. Springer-Verlag, 2000.

Temporal Reasoning

Belief extrapolation (or how to reason about observations and unpredicted change)

Florence Dupin de Saint-Cyr
LERIA,
Université d'Angers (France)
bannay@info.univ-angers.fr

Jérôme Lang
IRIT,
Université Paul Sabatier, Toulouse (France)
lang@irit.fr

Abstract

We give a logical framework for reasoning with observations at different time points. We call belief extrapolation the process of completing initial belief sets stemming from observations by assuming minimal change. We give a general semantic and we propose several extrapolation operators. We study some properties verified by these operators and investigate computational issues, including computational complexity. We study in detail the position of belief extrapolation with respect to revision and update ; in particular, we show that, although it deals with non-static worlds, belief extrapolation cannot be a belief update operator in the sense of Katsuno and Mendelzon; then, we show how to integrate update and extrapolation. Lastly, closely related lines of work are positioned with respect to belief extrapolation.

1 INTRODUCTION

The distinction between belief revision and update [12] is usually thought of this way: revision incorporates newly acquired beliefs about a static world while update modifies beliefs according to a change in the system, "notified" by an "input" formula φ . However, there is some ambiguity in the exact nature of the input formula φ . What does "a notification of the change" mean exactly? At least two distinct interpretations are possible:

- some time has passed between the initial time point t_0 when the initial belief state K was known to hold and a later time point t_1 when φ is *observed* (by means of a sensor, a request to a database, a notification by another agent, etc.);
- φ is the *expected consequence* of an action performed between t_0 and t_1.

Failing to distinguish these types of information may lead to counterintuitive results, as seen on the following example:

We consider a system to diagnose, whose components may fail independently, which means that there is no prior causal link between the failure of a component and the failure of another one. To simplify the problem we assume that for a given component, the transition from the failure mode to the working mode is not more exceptional than the transition from the working mode to the failure mode[1].

Scenario 1: *at* $t = 1$, *we know that exactly one of components a and b is faulty, and c works correctly:* $K = (faulty\_a \oplus faulty\_b) \wedge \neg faulty\_c$ *where* $\oplus$ *denotes exclusive or. We wait for some time, and then (at* $t = 2$*), some new observations (coming for instance from tests) make us learn that component b is ok but c is not:* $\varphi = \neg faulty\_b \wedge faulty\_c$.

Scenario 2: *at* $t = 1$, *as for Scenario 1,* $K = (faulty\_a \oplus faulty\_b) \wedge \neg faulty\_c$. *Then the action of repairing component b as well as the (unfortunate) action of breaking component c are performed. The effects of these two independent actions can be expressed by* $\varphi = \neg faulty\_b \wedge faulty\_c$.

These two scenarios are different and must lead to different conclusions. In case of Scenario 1, the intuitive way of changing beliefs is: "since component b is ok and changes are exceptional, there are good reasons to believe that b was already ok at time 1, so component a was the faulty one and most expectedly still is; as to c that has just been oberved faulty, there is no am-

[1]This assumption is not so irrealistic, considering that other agents can repair a faulty component without the "main" agent (the beliefs of whom we are interested in) being aware.

biguity". Therefore, the belief state at time 2 should be $faulty\_a \wedge \neg faulty\_b \wedge faulty\_c$. The case of Scenario 2 is different: the effect of repairing b does not bring any new information on its former state, nor, *a fortiori*, on the former state of a; the belief about a should thus remain unchanged and the belief state at $t = 2$ should be $\neg faulty\_b \wedge faulty\_c$.

Now, belief update may be the right thing to do for Scenario 2, but not for Scenario 1. The principle responsible to this inadequacy of belief update to handle Scenario 1 is the most "typical" property of belief update, namely U8 (see Section 5) which says that models of K should be updated independently; as a consequence, usual belief update operators satisfy $K \diamond \varphi = \neg faulty\_b \wedge faulty\_c$; this is the expected result for Scenario 2 but not for Scenario 1 (where we expect $faulty\_a \wedge \neg faulty\_b \wedge faulty\_c$).

From these two scenarios we may advance an hypothesis: *belief update is not appropriate when the input is an observation.* Two questions follow: (Q1) When is it appropriate? (Q2) How should we change a belief base to take account of observations occuring at different time points? As to Q1, Scenario 2 suggests that belief update works when *the input is the expected result (or projection) of an action (or an event) whose occurrence is known to the agent.* Q2 leads to defining a different operator that we call *belief extrapolation.* This operator takes a sequence of observations assumed to hold for certain and projects these observations forwards and backwards, assuming that fluents tend to persist throughout time. Such operators are not completely new (they have been considered in [3] and [17] – see the related work Section). The basic assumptions for extrapolation operators are the following two ones:

1. *the agent can only observe some properties at some time points*, but does not have the ability of performing actions; furthermore, if events occur, they are perceived by the agent through the observations only (for instance, the occurrence of the event that it will be raining tonight will be perceived by actually *seeing* the rain, or seeing the wet ground tomorrow morning, but not by listening to today's weather forecast). This is why changes can be qualified as *unpredicted.* In Section 5, however, we show that it is possible to generalize extrapolation operators so as to handle actions as well, provided that they are distinguished from observations, in order to minimize only the fluents not concerned by action affects.
2. *the system is inertial*, which means that by default, the system remains in a static state. This assumption justifies the use of a change minimization policy.

The rest of the paper is organized as follows. In Section 2 we give some background about revision and update. In Section 3 we formally define belief extrapolation; we propose several extrapolation operators and study their properties. In Section 4 we investigate computational issues, and propose a practical method for computing extrapolated beliefs for one of the formerly considered extrapolation operators. In Section 5 and 6, we study the connections between extrapolation and, respectively, revision (including iterated revision) and update. Namely, we show in Section 4 that under reasonable assumptions, extrapolation can be seen as a particular belief revision process where we revise the beliefs about the persistence of fluents by the time-stamped observations, and in Section 5 that a belief extrapolation operator cannot be a belief update operator unless very strong assumptions are made; rather, we show that extrapolation and update are complementary and that a full framework for reasoning about beliefs across time should involve both of them. This helps us assessing the respective positions of belief update and belief extrapolation in Sandewall's taxonomy of logically represented dynamical systems. Connections with several other works, including pioneering works on extrapolation [3] and [17], are investigated in Section 6.

2 BELIEF EXTRAPOLATION

2.1 SCENARIOS AND TRAJECTORIES

Let $\mathcal{L}$ be a propositional language built from a finite set of propositional variables $\mathcal{V} = \{v_1, ..., v_n\}$, the usual connectives, and the Boolean constants $\top$ (tautology) and $\bot$ (contradiction). $\mathcal{M} = 2^{\mathcal{V}}$ is the set of interpretations for $\mathcal{V}$. Formulas of $\mathcal{L}$ are denoted by Greek letters (φ, ψ etc.) and interpretations are denoted by m_i, m_j etc. If $\varphi \in \mathcal{L}$ then $Mod(\varphi)$ is the set of all models of φ. A *fluent* f is a literal, i.e., a variable or its negation; if $f = v_i$ (resp. $\neg v_i$) then $\neg f$ is $\neg v_i$ (resp. v_i). The set of fluents is $\mathcal{F} = \{v_1, \neg v_1, ..., v_n, \neg v_n\}$.

Definition 1 (scenarios)
A scenario $\Sigma = \langle \varphi_1, ..., \varphi_N \rangle$ *is a finite sequence of propositional formulas. We denote by* $\Sigma(t)$ *the* t^{th} *element* φ_t *of the sequence.* $|\Sigma|$ *denotes the length of* Σ. $\mathcal{S}$ *is the set of all scenarios.*
Σ *is said to be* consistent *if and only if for each* $t \in 1..|\Sigma|$, $\Sigma(t)$ *is consistent.*

For any two scenarios Σ, Σ' *of length* N, *we define:* $\Sigma \models \Sigma'$ *if and only if* $\forall t$, $\Sigma(t) \models \Sigma'(t)$;

$\Sigma \equiv \Sigma'$ *if and only if* $\Sigma \models \Sigma'$ *and* $\Sigma' \models \Sigma$;
$\Sigma \wedge \Sigma' = \langle \Sigma(t) \wedge \Sigma'(t) | t \in 1 \ldots N \rangle$.

The concatenation $\Sigma.\Sigma'$ *of two scenarios* Σ *and* Σ', *is defined by:*

$$(\Sigma.\Sigma')(t) = \begin{cases} \Sigma(t) & \text{if } t \leq |\Sigma| \\ \Sigma'(t - |\Sigma|) & \text{if } t > |\Sigma| \end{cases}$$

The scenario Σ *s.t.* $\Sigma(t) = \varphi$ *and* $\Sigma(t') = \top$ *for all* $t' \neq t$ *is denoted by* $[t]\varphi$.

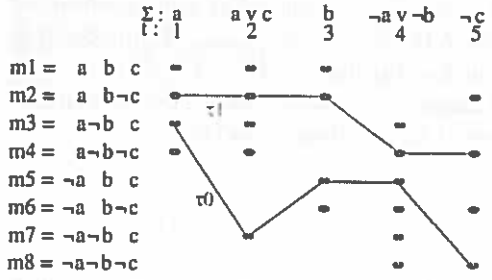
Definition 2 (trajectories)
A trajectory *of length* N *is a sequence* $\tau = \langle \tau(1), .., \tau(N) \rangle$ *of interpretations (i.e.,* $\tau(i) \in \mathcal{M}$*).* $TRAJ_N$ *denotes the set of all trajectories of length* N.

If Σ *is a scenario and* τ *a trajectory of the same length, we say that* τ satisfies Σ *(denoted by* $\tau \models \Sigma$*) if and only if* $\forall t \in 1..N$ *we have* $\tau(t) \models \Sigma(t)$.

$Traj(\Sigma) = \{\tau \in TRAJ_{|\Sigma|} | \tau \models \Sigma\}$ *is the set of trajectories satisfying* Σ.

τ *is* static *if and only if* $\tau(1) = ... = \tau(N)$.

Example 1 *Let* $\Sigma = \langle a, a \vee c, b, \neg a \vee \neg b, \neg c \rangle$ *and the possible interpretations* $m1, \ldots, m8$ *as on the figure below. Many trajectories satisfy* Σ, *for instance* $\tau_0 = \langle m3, m7, m5, m5, m8 \rangle$ *and* $\tau_1 = \langle m2, m2, m2, m4, m4 \rangle$.



Note that there is no static trajectory satisfying Σ.

Definition 3 (change set)
The change set $Ch(\tau)$ *of a trajectory* τ *is defined by:*

$$Ch(\tau) = \left\{ \langle f, t \rangle \;\middle|\; \begin{array}{l} f \in \mathcal{F},\ t \in 2..|\tau|, \\ \tau(t-1) \models \neg f \text{ and } \tau(t) \models f \end{array} \right\}$$

We also define $Ch(\tau, f) = \{t \mid \langle f, t \rangle \in Ch(\tau)\}$.

Example 1, continued
$Ch(\tau_0) = \{\langle \neg a, 2 \rangle, \langle b, 3 \rangle, \langle \neg b, 5 \rangle, \langle \neg c, 5 \rangle\}$ *and*
$Ch(\tau_1) = \{\langle \neg b, 4 \rangle\}$.
We see that τ_1 *looks "more static" than* τ_0.

The input of an extrapolation operator is a consistent scenario Σ. Formulas of Σ are *observations* (coming from sensors, from communication with other agents etc.). For the sake of simplicity we do not consider static laws expressing dependence between fluents; however, it is possible (though not optimal in practice) to give these static laws the status of observations: at each time point the initial belief will thus be the conjunction of the observation and the static laws.

The consistency of Σ means that observations are assumed to be reliable (see [10] for a discussion on this point).

2.2 PREFERRED TRAJECTORIES AND EXTRAPOLATION OPERATORS

Given an observation scenario Σ, *belief extrapolation* consists in completing it using persistence assumptions (such a process is referred to as *chronicle completion* in [19]). The rationale of belief extrapolation is that as long as nothing tells the contrary, fluents do not change. So, our aim is to find the best trajectories satisfying the observations, i.e., the most static trajectories. This is why we introduce preference relations on trajectories:

Definition 4 (preference relations on trajectories)
A preference relation $\preceq$ *is a reflexive and transitive relation on* $TRAJ_N$ *(not necessarily connected). We note*

- $\tau \prec \tau'$ *(read "*τ *is strictly preferred to* τ'*") for* $(\tau \preceq \tau'$ *and not* $\tau' \preceq \tau)$,
- $\tau \sim \tau'$ *for* $(\tau \preceq \tau'$ *and* $\tau' \preceq \tau)$.

Let $X \subseteq TRAJ_N$, *a trajectory* $\tau \in X$ *is* minimal *w.r.t.* $\preceq$ *in* X *if and only if there is no* τ' *in* X *such that* $\tau' \prec \tau$.

Definition 5
- $\preceq$ *is said* inertial *if and only if*
  1. *for any two static trajectories of same length* τ *and* τ' *we have* $\tau \sim \tau'$ *and*
  2. *for any static* τ *and any non-static* τ' *(of same length) we have* $\tau \prec \tau'$.
- $\preceq$ *is said to be* change-based *if and only if there exists a reflexive and transitive relation* $\preceq^c$ *on* $\mathcal{F} \times \{1, ..., N\}$ *such that for any two trajectories* τ, τ' *we have* $\tau \preceq \tau'$ *if and only if* $Ch(\tau) \preceq^c Ch(\tau')$.

Inertia says that if there exist static trajectories for a given scenario Σ, then the set of preferred trajectories must be the set of all static trajectories for Σ; it means that if it can be consistently assumed that no change occured, then completion consists in jumping to this conclusion.

The fact that a preference relation is change-based induces a simplification, because the relation is then easier to specify. Intuitive preference relations are most of the time change-based (and actually, almost all examples of preference relations given further on are change-based). Example of preference relation which are not change-based are complex relations where the *context* of the change is relevant (see further).

We are now in position to formally define an extrapolation operator.

Definition 6 (extrapolation operators)
Any preference relation $\preceq$ induces an extrapolation operator $\updownarrow_{\preceq}: \mathcal{S} \rightarrow \mathcal{S}$ defined by:

$$Mod(\Sigma\updownarrow_{\preceq}(t)) = \{\tau(t) \mid \tau \in Min(\preceq, Traj(\Sigma))\}$$

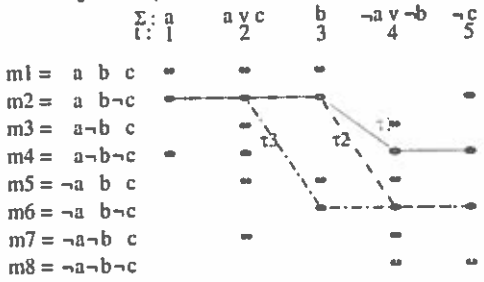
2.3 SOME EXAMPLES OF PREFERENCE RELATIONS

We now give several examples of preference relations. The first one just consists in minimizing the number of elementary changes throughout the process.

- number of changes:

$$\tau \preceq_{nc} \tau' \text{ if and only if } |Ch(\tau)| \leq |Ch(\tau')|$$

Example 1 (continued)



The three trajectories minimal for $\preceq_{nc}$ are:

trajectory τ	$Ch(\tau)$
$\tau_1 = \langle m2, m2, m2, m4, m4\rangle$	$\{\langle \neg b, 4\rangle\}$
$\tau_2 = \langle m2, m2, m2, m6, m6\rangle$	$\{\langle \neg a, 4\rangle\}$
$\tau_3 = \langle m2, m2, m6, m6, m6\rangle$	$\{\langle \neg a, 3\rangle\}$

Extrapolating Σ w.r.t. $\preceq_{nc}$ gives:

$\Sigma\updownarrow_{\preceq_{nc}} = \langle a \wedge b \wedge \neg c,\ a \wedge b \wedge \neg c,\ b \wedge \neg c,\ (a \oplus b) \wedge \neg c,\ (a \oplus b) \wedge \neg c\rangle$;

If the number of changes is minimized fluent by fluent instead of globally, we get the following preference relation:

- number of changes by fluent:

$$\tau \preceq_{ncf} \tau' \text{ iff } \forall f \in \mathcal{F},\ |Ch(\tau, f)| \leq |Ch(\tau', f)|$$

While the latter two preference relations were insensitive to the time points when the fluents changed, the next one is not:

- change set inclusion:

$$\tau \preceq_{csi} \tau' \text{ if and only if } Ch(\tau) \subseteq Ch(\tau')$$

We may also minimize the set of fluents that change at least once:

- inclusion of changing fluents:

$$\tau \preceq_{icf} \tau' \text{ if and only if}$$
$$\{f \in \mathcal{F} | Ch(\tau, f) \neq \emptyset\} \subseteq \{f \in \mathcal{F} | Ch(\tau', f) \neq \emptyset\}$$

A refinement of the global minimzation of the number of changes consists in attaching a penalty to each possible change. The simplest assumption is when this penalty depends only on the identity of the fluent involved in the change. In this case, let $k : \mathcal{F} \rightarrow \mathbb{N}^*$, where $k(f)$ is the penalty induced by a change from $\neg f$ to f, and let $K(\tau) = \sum_t \sum_{(f,t) \in Ch(\tau)} k(f)$.Note that we do not necessarily have $k(f) = k(\neg f)$ since some fluents can change more easily than other, for instance $k(alive)$ should be less than $k(\neg alive)$[2].

- penalty:

$$\tau \preceq^k_{cp} \tau' \text{ if and only if } K(\tau) \leq K(\tau')$$

Note that we recover $\preceq_{nc}$ by letting $k(f) = 1$ for all f.

A completely different type of minimization, which has been proposed in many places before, consists in delaying change as much as possible, e.g., preferring trajectories where change occur as late as possible. This policy is called *chronological minimization* in [20]; it is extensively discussed in [19].

- chronological : the preference relation $\preceq_{achr}$ is defined recursively by

[2] As proposed by a reviewer, it could appear more natural to allow no-cost changes, i.e., defining k as a function from $\mathcal{F}$ to $\mathbb{N}$ (instead of $\mathbb{N}^*$). The reason why we impose that changes have a strictly positive penalty is to impose inertia for each fluent. Non-inertial extrapolation would deserve another paper (see some comments in Conclusion).

$$\tau \preceq_{chr} \tau' \text{ if and only if}$$
$$\begin{cases} p = 1 \\ \text{or } Ch(\tau,1) \subset Ch(\tau',1) \\ \text{or } (Ch(\tau,1) = Ch(\tau',1) \\ \quad \text{and } \langle \tau(2)...\tau(p)\rangle \preceq_{chr} \langle \tau'(2)..\tau'(p)\rangle) \end{cases}$$

Alternatively, one may wish to prefer earlier changes:

- anti-chronological :

$$\tau \preceq_{achr} \tau' \text{ if and only if}$$
$$\begin{cases} p = 1 \\ \text{or } Ch(\tau,p) \subset Ch(\tau',p) \\ \text{or } (Ch(\tau,p) = Ch(\tau',p) \\ \quad \text{and } \langle \tau(1)..\tau(p-1)\rangle \preceq_{achr} \langle \tau'(1)...\tau'(p-1)\rangle) \end{cases}$$

Lastly, we may want to compare *sequences of changes* independently of their precise location in time: only the changes involved and the order in which they occured are relevant.

- change sequence dominance:
$\tau \preceq_{cseq} \tau'$ if and only if there is an injective mapping F from $Ch(\tau)$ to $Ch(\tau')$ satisfying

1. fluent preservation: $\langle f', t'\rangle = F(\langle f, t\rangle)$ implies $f' = f$;
2. monotonicity w.r.t. time: for any two changes $\langle f_1, t_1\rangle$, $\langle f_2, t_2\rangle \in Ch(\tau)$, if $\langle f_1', t_1'\rangle = F(\langle f_1, t_1\rangle)$ and $\langle f_2', t_2'\rangle = F(\langle f_2, t_2\rangle)$ then $t_1 < t_2$ if and only if $t_1' < t_2'$.

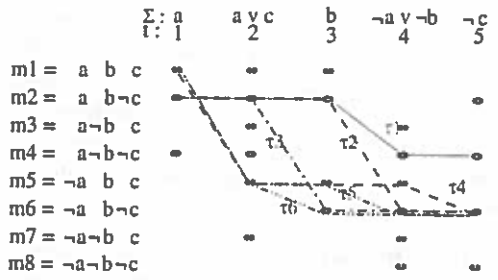
Note that $\tau \sim_{cseq} \tau'$ if and only if the fluents which change in τ and in τ' are the same *and strictly in the same order.*

Example 1 (continued)
The three trajectories minimal for $\preceq_{icf}$ are the same than for $\preceq_{nc}$: τ_1, τ_2 and τ_3.

There are three more minimal trajectories for $\preceq_{csi}$, namely:

trajectory τ	$Ch(\tau)$
$\tau_4 = \langle m1, m5, m5, m5, m6\rangle$	$\{\langle \neg a, 2\rangle, \langle \neg c, 5\rangle\}$
$\tau_5 = \langle m1, m5, m5, m6, m6\rangle$	$\{\langle \neg a, 2\rangle, \langle \neg c, 4\rangle\}$
$\tau_6 = \langle m1, m5, m6, m6, m6\rangle$	$\{\langle \neg a, 2\rangle, \langle \neg c, 3\rangle\}$

As to $\preceq_{cseq}$, the minimal trajectories are τ_1, τ_2 and τ_3. Notice that $\tau_2 \prec_{cseq} \tau_4$ and that $\tau_4 \sim_{cseq} \tau_5 \sim_{cseq} \tau_6$. Remark also that $\tau_7 = \langle m1, m1, m6, m6, m6\rangle$ cannot be compared w.r.t. $\sim_{cseq}$ to τ_5 and τ_6: indeed, $Ch(\tau_7) = \{\langle \neg a, 3\rangle, \langle \neg c, 3\rangle\}$, therefore, in τ_7 the changes from $\neg a$ to a and from c to $\neg c$ occur simultaneously, whereas for τ_4, τ_5 and τ_6, the changes from $\neg a$ to a occurs before the change from c to $\neg c$.



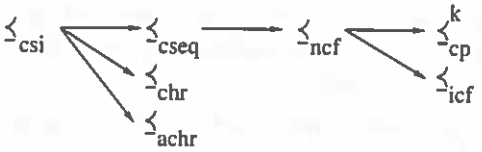
Abbreviating $\updownarrow_{\preceq_x}$ by $\updownarrow_x$ we have:
$\Sigma\updownarrow_{nc} = \Sigma\updownarrow_{ncf} = \Sigma\updownarrow_{cseq} = \langle a \wedge b \wedge \neg c,\ a \wedge b \wedge \neg c,\ b \wedge \neg c,\ (a \oplus b) \wedge \neg c,\ (a \oplus b) \wedge \neg c\rangle$;

$\Sigma\updownarrow_{csi} = \langle a \wedge b,\ (a \oplus c) \wedge b,\ b \wedge (\neg a \vee \neg c),\ (\neg a \wedge b) \vee (a \wedge \neg b \wedge \neg c),\ (a \oplus b) \wedge \neg c\rangle$;

$\Sigma\updownarrow_{chr} = \langle a \wedge b \wedge c,\ a \wedge b \wedge c,\ a \wedge b \wedge c,\ (a \oplus b) \wedge \neg c,\ (a \oplus b) \wedge \neg c\rangle$;

$\Sigma\updownarrow_{achr} = \langle a \wedge b,\ (a \oplus c) \wedge b,\ \neg a \wedge b \wedge \neg c,\ \neg a \wedge b \wedge \neg c,\ \neg a \wedge b \wedge \neg c\rangle$.

Note that we have the following inclusions between preference relations, represented by this Hasse diagram (edges represent inclusion).



All these preference relations consider atomic changes as independent. It may be the case, however, that fluents do not change independently from each other. This is the case when changes are caused by *events* as in [4]. A (deterministic) *event* ε is a mapping from $\mathcal{M}$ to $\mathcal{M}$, where $\varepsilon(m)$ is the result of event ε when the initial state is m. The *empty event* $\varepsilon_\emptyset$ is the identity. Each event has a penalty $k(\varepsilon) \geq 0$ with $k(\varepsilon) = 0$ if and only if $\varepsilon = \varepsilon_\emptyset$.

Given a set of events $\mathcal{E}$ and two models m, m', let $k^{\mathcal{E}}(m,m') = \min\{k(\varepsilon) \mid \varepsilon \in \mathcal{E}, m' = \varepsilon(m)\}$ ($= +\infty$ if no such sequence exists) and $K^{\mathcal{E}}(\tau) = \sum_{t=0..N-1} k^{\mathcal{E}}(\tau(t), \tau(t+1))$. We now define the event-based preference relation:

event penalty

$$\tau \preceq_e^{\mathcal{E}} \tau' \text{ if and only if } K^{\mathcal{E}}(\tau) \leq K^{\mathcal{E}}(\tau')$$

This definition can be refined by assuming that several consecutive event occurences are possible between two time points. This leads us to define an *event sequence* as a combination of events $\theta = \varepsilon_1 \circ ... \circ \varepsilon_p$, such that,

$\theta(m) = \varepsilon_1(\varepsilon_2(...(\varepsilon_p(m))))$; we let $k(\theta) = \sum_{i \in 1..p} k(\varepsilon_i)$. Given a set of events $\mathcal{E}$ and two models m, m', we now let $k^{\mathcal{E}}(m, m') = \min_{\theta | \theta(m)=m'} k(\theta)$ ($= +\infty$ if no such sequence exists). $K^{\mathcal{E}}(\tau)$ and $\preceq_e^{\mathcal{E}}$ are then defined as above. The events (or event sequences) minimizing $k^{\mathcal{E}}(m, m')$ can be considered as the plausible explanations of the transition from m to m'.

Example 1 (continued) *Let $\mathcal{E} = \{\varepsilon_\emptyset, \varepsilon_1, eps_2\}$ where ε_1 switches simulatenously the truth values of a and b: $\varepsilon_1(a, b, c) = (\neg a, \neg b, c)$; $\varepsilon_1(a, b, \neg c) = (\neg a, \neg b, \neg c)$; $\varepsilon_1(a, \neg b, c) = (\neg a, b, \neg c)$ etc. and ε_2 switches the truth value of c. Assume that $k(\varepsilon_1) = k(\varepsilon_2) = 1$. In both cases where we allow for single events only or event sequences, the only preferred trajectory is $\langle a \wedge b \wedge \neg c, a \wedge b \wedge \neg c, a \wedge b \wedge \neg c, \neg a \wedge \neg b \wedge \neg c, \neg a \wedge \neg b \wedge \neg c \rangle$, corresponding to the occurrence of event ε_1 between $t = 3$ and $t = 4$.*

Note that since the effects of events generally depend on the initial state, the data of $Ch(\tau)$ and $Ch(\tau')$ are generally not enough to decide whether $\tau \preceq_e^{\mathcal{E}} \tau'$, *i.e.*, event-based preference relations are generally not change-based.

2.4 PROPERTIES

We start by giving a set of basic properties and then we show that under weak conditions, they are satisfied by extrapolation operators.

Let Σ be a consistent scenario and φ, ψ two propositional formulas.

- **E1** $\Sigma\updownarrow \models \Sigma$;
- **E2** $\Sigma\updownarrow$ is consistent;
- **E3** if $\Sigma \equiv \Sigma'$ then $\Sigma\updownarrow \equiv \Sigma'\updownarrow$;
- **E4** $\Sigma\updownarrow(t) \wedge \varphi \models (\Sigma \wedge [t]\varphi)\updownarrow(t)$;
- **E5** if $\bigwedge_{t' \in 1..N} \Sigma(t')$ is consistent then for all $t \in 1 \ldots N$, we have $\Sigma\updownarrow(t) = \bigwedge_{t' \in 1..N} \Sigma(t')$;
- **E6** if $\Sigma\updownarrow(t) \wedge \varphi$ is consistent then $(\Sigma \wedge [t]\varphi)\updownarrow(t) \models \Sigma\updownarrow(t) \wedge \varphi$

Proposition 1

- *Any operator $\updownarrow_\preceq$: $\mathcal{S} \to \mathcal{S}$ induced by an inertial preference relation $\preceq$ satisfies the properties E1-E5;*
- *Any operator $\updownarrow_\preceq$: $\mathcal{S} \to \mathcal{S}$ induced by an inertial, total preference relation $\preceq$ satisfies E1-E6.*

E1 says that observations are reliable. E4 says that extrapolation is monotonic when adding a new observation consistent with the extrapolation of the former observations; E4 and E6 together mean that taking account of this new observation consistent with $\Sigma\updownarrow$ before or after extrapolation gives equivalent results. E5 means that if it can be consistently assumed that nothing changed, then this assumption is made, and therefore the extrapolated belief at each time point is the conjunction of all observations. Note that E1, E2 and E4 entail $\Sigma\updownarrow\updownarrow \equiv \Sigma\updownarrow$, *i.e.*, extrapolation is idempotent.

Beyond these rather undebatable properties, there are other intuitive properties that extrapolation operators may be required to satisfy:

- **E7** if $\varphi \models \psi$ or $\psi \models \varphi$ then $\forall t \neq |\Sigma|+1, |\Sigma|+2$, $(\Sigma.\varphi.\psi.\Sigma')\updownarrow(t) \equiv (\Sigma.(\varphi \wedge \psi).(\varphi \wedge \psi).\Sigma')\updownarrow(t)$;
- **E8** $\forall t \neq |\Sigma|+1$, $(\Sigma.\top.\Sigma')\updownarrow(t) \equiv \begin{cases} (\Sigma.\Sigma')\updownarrow(t) & \text{if } t \leq |\Sigma|+1 \\ (\Sigma.\Sigma')\updownarrow(t-1) & \text{if } t > |\Sigma|+1 \end{cases}$
- **E9** if $(\Sigma.\varphi)\updownarrow(|\Sigma|+1) \equiv (\varphi.\Sigma')\updownarrow(1)$ then $\forall t$,

$$(\Sigma.\varphi.\Sigma')\updownarrow(t) \equiv \begin{cases} (\Sigma.\varphi)\updownarrow(t) & \text{if } t \leq |\Sigma|+1 \\ (\varphi.\Sigma')\updownarrow(t-|\Sigma|) & \text{if } t > |\Sigma|+1 \end{cases}$$

Unlike properties E1-E6, which did not explicitely refer to the flow of time, E7-E8 express important properties related to persistence or independence of changes.

E7 says that if an observation is followed or preceded by a stronger information, then the extrapolated beliefs on other time points would have been the same, had we got twice the stronger information instead. Importantly, this property considers persistence not only forward but also backward: if an observation holds at a given time point, then by default it holds afterwards as well as before.

E8 means that an empty observation has no influence on extrapolated beliefs. An extrapolation operator violating E8 is the (weird) operator based on $\preceq$ s.t. $\tau \preceq \tau'$ if and only if $\forall t \geq 2$, $\{f | \langle f, t \rangle \in Ch(\tau)$ or $\langle f, t+1 \rangle \in Ch(\tau)\} \subseteq \{f | \langle f, t \rangle \in Ch(\tau')$ or $\langle f, t+1 \rangle \in Ch(\tau')\}$.

E9 is a kind of Markov property and implies that an extrapolation problem can sometimes be decomposed into independent subproblems; especially, the condition is verified as soon as there is a *complete* observation (having a single model) at $t = |\Sigma|+1$.

Proposition 2

- *$\updownarrow_{nc}$ and $\updownarrow_{cp}^k$ satisfy E1-E9.*
- *$\updownarrow_{csi}$, $\updownarrow_{cseq}$ and $\updownarrow_{ncf}$ satisfy E1-E5 and E7-E9.*
- *$\updownarrow_{chr}$ and $\updownarrow_{achr}$ satisfy E1-E5, E8 and E9.*
- *$\updownarrow_{icf}$ satisfies E1-E5, E7 and E8.*

- $\updownarrow_e^E$ *satisfies E1-E6, E8 and E9.*

Another interesting property is the *reversibility* of extrapolation. We first need to define the *reversal* of a scenario or a trajectory.

Definition 7
For any scenario $\Sigma = \langle \Sigma(1), .., \Sigma(N) \rangle$, *we define the reversal of* Σ *by*

$$Reverse(\Sigma) = \langle \Sigma(N), .., \Sigma(1) \rangle$$

For any trajectory τ *we define the reversal of* τ *by*

$$Reverse(\tau)(t) = \tau(N - t)$$

Now, we introduce the *reversibility* property:

(R) $Reverse(\Sigma\updownarrow) = (reverse(\Sigma))\updownarrow$.

Proposition 3 $\updownarrow_\preceq$ *satisfies (R) if and only if* $\preceq$ *is such that for all* τ, τ' *of same length* N *we have* $\tau \preceq \tau'$ *if and only if* $Reverse(\tau) \preceq Reverse(\tau')$

Proposition 4

- $\updownarrow_{csi}, \updownarrow_{ncf}, \updownarrow_{nc}$ *and* $\updownarrow_{icf}$ *satisfy (R);*
- $\updownarrow_{cp}^k$ *satisfies (R) if and only if* $\forall f \in \mathcal{F}$ *we have* $k(f) = k(\neg f)$;
- $\updownarrow_{chr}$ *and* $\updownarrow_{achr}$ *do not satisfy (R);*
- $\updownarrow_{ep}^E$ *satisfies (R) if and only if* $\forall m, m'$ *we have* $k^{\mathcal{E}}(m, m') = k^{\mathcal{E}}(m', m)$

3 COMPUTATIONAL ISSUES

3.1 COMPLEXITY

We start by identifying the complexity of belief extrapolation for each of the preference relations given above. We assume that the reader is familiar with the complexity classes Δ_2^p, Π_2^p of the polynomial hierarchy. (Since this is not crucial for the rest of the paper, we do not give any details. The reader can refer e.g. [18]).

Proposition 5 EXTRA$_x$ *is the decision problem: given* Σ, ψ *and* t, *decide whether* $\Sigma\updownarrow_x(t) \models \psi$ *holds. Then we have the following:* EXTRA$_{csi}$, EXTRA$_{icf}$, EXTRA$_{cseq}$, EXTRA$_{chr}$ *and* EXTRA$_{achr}$ *are* Π_2^p*-complete;* EXTRA$_{cp}^k$ *is in* Δ_2^p [3]; EXTRA$_{nc}$ *is in* $\Delta_2^p[\mathcal{O}(\log n)]$.

[3]This result has to be drawn together with the complexity of inference under the backwards semantics in [17] since both problems are closely related (see Section 6).

3.2 COMPUTING EXTRAPOLATED BELIEFS

For the rest of this Section we fix the preference relation to $\preceq_{csi}$.

We define now a *minimal explanation* γ for Σ as the change set $Ch(\tau)$ of a $\preceq_{csi}$-preferred trajectory τ of Σ.

The information provided by the set of all minimal explanations is very rich: even if it does not always determine the preferred trajectories in the general case (because there may be several trajectories having the same change set), it does as soon as there is a t_i such that $\Sigma(t_i)$ is *complete* (i.e., has only one model m_i), and in this case, from the set of minimal explanations together with m_i we can decide in polynomial time whether $\Sigma\updownarrow(t) \models \psi$ holds for any t and ψ.

Given a scenario Σ (of length N), the number of minimal explanations can be very large, even when the number of changes is small:

Example 2 $\Sigma = \langle a \wedge b \wedge c \wedge d,\ \top,\ \neg b,\ d,\ \neg a,\ \top,\ b,\ \neg c \vee \neg d \rangle$ *has exactly* $4 \times 2 \times 4 \times (7+4) = 352$ *minimal explanations (and as many minimal trajectories). Now, these 352 minimal explanations can be expressed compactly:* γ *is a minimal explanation for* Σ *if and only if it contains (i)* $\langle \neg a, t \rangle$ *for (exactly) one* $t \in \{2..5\}$ *(meaning that a became false exactly once, at 2, 3, 4 or 5 – since it was true at 1 and false at 5), (ii)* $\langle \neg b, t \rangle$ *for one* $t \in \{2, 3\}$, $\langle b, t \rangle$ *for one* $t \in \{4..7\}$, *and either (iii)* $\langle \neg c, t \rangle$ *for one* $t \in \{2..8\}$ *or (iii')* $\langle \neg d, t \rangle$ *for one* $t \in \{5..8\}$ *(and nothing else).*

To capture this simple notion we introduce the following notion: a *compact change set* Γ is a set $\{\langle f_i, t_i^-, t_i^+ \rangle \mid i = 1..p\}$ where the f_i are fluents and the (t_i^-, t_i^+) are pairs of time points such that $0 < t_i^- \leq t_i^+ \leq N$. The set of change sets covered with Γ is $Cov(\Gamma) = \{\langle f_1, t_1 \rangle, ..., \langle f_p, t_p \rangle \mid \forall i = 1..p, t_i^- \leq t_i \leq t_i^+\}$. We say that Γ *explains* Σ if and only if every $\gamma \in Cov(\Gamma)$ explains Σ.

Intuitively, a compact change set consists of the disjunction of all the change sets it covers. We say that compact change set Γ *explains minimally* Σ if and only if $\forall \gamma \in Cov(\Gamma)$, γ is a minimal explanation for Σ.

Γ is a *maximally compact minimal explanation* (MCME) for Σ if and only if (i) Γ explains minimally Σ and (ii) there is no Γ' explaining minimally Σ such that $Cov(\Gamma') \subset Cov(\Gamma)$. The set of MCMEs for Σ expresses minimal explanations in a compact way and is redundancy-free: indeed, it can be easily shown that for each minimal explanation γ for Σ there is one and only one MCME Γ for Σ covering γ. In the previous

example, there are two MCMEs:
$\{\langle\neg a,2,5\rangle,\langle\neg b,2,3\rangle,\langle b,4,7\rangle,\langle\neg c,2,8\rangle\}$ and
$\{\langle\neg a,2,5\rangle,\langle\neg b,2,3\rangle,\langle b,4,7\rangle,\langle\neg d,5,8\rangle\}$.

When observations are disjunction-free we easily get:

Proposition 6 *If for each t, $\Sigma(t)$ is a consistent conjunction of fluents, then there is only one maximally compact minimal explanation for Σ.*

Compact minimal explanations can be computed by adding to Σ a maximal set of persistence assumptions expressing that each fluent f persists by default during intervals between two consecutive observations "relevant" to f. This notion of relevance is modeled in this simple way (see [14]): We say that an observation φ is *relevant* to a propositional variable v, denoted $Rel(\varphi, f)$, if and only if there is no formula equivalent to φ in which v does not appear at all[4].

Now, for a variable v, let $R_\Sigma(v) = \{t \in 1..N \text{ s.t. } Rel(\Sigma(t), v)\}$; now, the set of *relevant intervals* $RI_\Sigma(v)$ to v w.r.t. Σ is defined by $RI_\Sigma(v) = \{[t,t'] \mid t \in R_\Sigma(v), t' \in R_\Sigma(v), [t+1,t'-1] \cap R_\Sigma(v) = \emptyset\}$. The intuitive meaning of relevant intervals is that it is sound to consider only persistence assumptions regarding a given variable within these intervals only[5].

Variables are then time-stamped by the time point they refer to[6]: if $v \in \mathcal{V}$ and $t \in 1..N$ then v_t holds if and only if v holds at time t. If ϕ is a formula built on $\mathcal{V}$ then ϕ_t is obtained from ϕ by replacing each variable v in ϕ by v_t. Let $\Sigma^* = \{\Sigma(t)_t \mid t = 1..N\}$ and let the set of *persistence assumptions* for Σ be $P_\Sigma = \bigcup_{\substack{v \in \mathcal{V} \\ [t,t'] \in RI_\Sigma(v)}} \{v_t \rightarrow v_{t'}, \neg v_t \rightarrow \neg v_{t'}\}$. Define the *P-explanations* for $\langle P_\Sigma, \Sigma^*\rangle$ as the minimal subsets π of P_Σ such that $P_\Sigma \setminus \pi$ is consistent with Σ^*. Each P-explanation π can be identified with a compact change set $\Gamma(\pi)$ in an obvious way; for instance, $\pi = \{a_4 \rightarrow a_7, \neg c_5 \rightarrow \neg c_6\}$ draws $\Gamma(\pi) = \{\langle\neg a,5,7\rangle,\langle c,6,6\rangle\}$, i.e., a became false once between 4 and 7 (i.e., at 5, 6, or 7) and c became true at 6. We have the following:

Proposition 7

- *for each P-explanation π for $\langle P_\Sigma, \Sigma^*\rangle$, each change set $\gamma \in \Gamma(\pi)$ is a minimal explanation for Σ* [7];
- *each minimal explanation for Σ is covered by exactly one P-explanation, i.e., there exists one and only one π such that $\gamma \in \Gamma(\pi)$.*

Thus, a covering set of compact explanations can be computed using abduction algorithms. We also point that complexity results and tractable classes about logic-based abduction [9], can be used to evaluate the difficulty of this task and find some tractable subclasses of extrapolation.

4 EXTRAPOLATION AND REVISION

We do not recall here the well-known AGM postulates [1], which are anyway not crucial to the understanding of this Section.

Belief revision usually deals with static worlds, and therefore works with formulas pertaining to the same time point. However, as remarked in [11], "what is important for revision is not that the world is static, but that the propositions used to describe the world are static". This explains why, when indexing propositional variables by time points as we do in the Section 3, we manage to capture a fully dynamic process within a pure belief revision framework: belief extrapolation is equivalent to revising the prior belief that all fluents persist throughout time by the observations.

4.1 FROM EXTRAPOLATION TO REVISION

A first connection appears when we consider sequences of *two* observations only. Extrapolation operators on sequences of length 2 are called *2-extrapolation* operators.

Since the language is finite, we consider revision operators as acting on propositional formulas as in [13] (and not on belief sets). From $\updownarrow: \mathcal{L}^2 \rightarrow \mathcal{L}^2$ we induce the belief change operator $\star = \Phi(\updownarrow) : \mathcal{L}^2 \rightarrow \mathcal{L}^2$ by $KB \star \alpha = \langle KB, \alpha\rangle \updownarrow (2)$. Conversely, from a belief change operator $\star$ we induce the operator $\updownarrow = \Psi(\star)$ by $\langle\Sigma(1),\Sigma(2)\rangle \updownarrow (2) = \Sigma(1) \star \Sigma(2)$ and $\langle\Sigma(1),\Sigma(2)\rangle \updownarrow (1) = \Sigma(2) \star \Sigma(1)$. Note that, obviously, $\Phi(\Psi(\star)) = \star$, but $\Psi(\Phi(\updownarrow)) = \updownarrow$ does not necessarily hold.

[4] or equivalently, φ is *not* relevant to v if and only if $\varphi_{v \leftarrow \top}$, $\varphi_{v \leftarrow \bot}$ and φ are logically equivalent.

[5] This method could be further improved by distinguishing between *positive* and *negative* observations.

[6] A similar translation in used in [11] so as to represent beliefs about different time point in a pure belief revision framework.

[7] Note that it is not always the case that the compact explanations Γ computed according to the previous method are *maximally* compact, as seen on the following example: $\Sigma = \langle a, c, a \vee c, b, \neg a \vee b, \neg a\rangle$ yields the compact minimal explanations $\{\langle\neg a,4,5\rangle\}$ and $\langle\neg a,6,6\rangle\}$ which are not *maximally* compact since their union $\{\langle a,4,6\rangle\}$ is a MCME. If we wish to obtain the MCMEs for Σ we have to define a compactification procedure whose precise description is outside the scope of the paper.

Proposition 8

1. $\ddagger$: $\mathcal{L}^2 \to \mathcal{L}^2$ *satisfies E1-E6 if and only if* $\Phi(\ddagger)$ *is an AGM belief revision operator;*
2. $\star$ *is an AGM belief revision operator if and only if* $\Psi(\star)$ *is a 2-extrapolation operator satisfying E1-E6.*

This result sheds more light on what belief extrapolation intuitively consists of. Assuming $\preceq$ c! connected and $N = 2$, the extrapolated belief set at time 2 consists of the revision of beliefs at time 1, which we believe to persist to time 2, by the observations at time 2 which hold for certain. This looks like a pure revision framework, but have in mind that the world is dynamic. Symmetrically, the extrapolated belief set at time 1 consists of the revision of beliefs at time 2, which we believe to " persist backwards" to time 1, by the observations at time 1.

4.2 FROM REVISION TO EXTRAPOLATION

Proposition 9 *Let* $\star$ *be an AGM belief revision operator defined on a propositional language generated by* $\mathcal{V} = \{v_1, .., v_n\}$. *We index variables by time points as in Section 3. Let* $PERS = \bigwedge_{t=1..N-1}^{v \in \mathcal{V}} v_t \leftrightarrow v_{t+1}$ *and* $\Sigma^* = \{\Sigma(t)_t \mid t = 1..N\}$. *Then the operator* $E : \Sigma \mapsto E(\Sigma)(t) = \psi_t$ *where* ψ_t *is the strongest formula mentioning only variables indexed by* t *such that* $PERS \star \Sigma^* \models \psi_t$, *is an extrapolation operator satisfying E1-E6.*

Thus, we can draw an extrapolation operator from any revision operator. Actually, it can be shown that properties E1-E6 correspond to a reformulation of the AGM postulates so as to take account from different time points; however, they are only little relevant to the specificities concerned with the flow of time, such as forms of Markovianity and independence of changes expressed by E7-E9. Therefore, as soon as we require other properties than the basic ones, extrapolation is "more specific" than AGM revision.

4.3 EXTRAPOLATION AND ITERATED REVISION

Now, an interesting question is the possible connections between extrapolation and *iterated belief revision*. Especially, Lehmann's iterated revision [16] considers sequences of consistent formulas[8], therefore the input is exactly as ours. Since he is only concerned with the final result of the iterated revision process, what we can compare given a scenario Σ is the belief resulting from Σ, denoted by $[\Sigma]$, with the projection of the extrapolation at the final time point $\Sigma\ddagger(N)$ (where $N = |\Sigma|$).

Proposition 10 *Let* $\ddagger$ *be a basic extrapolation operator. The operator* $\Sigma \mapsto \Sigma\ddagger(N)$ *satisfies postulates I1, I2, I3, I5 and I6 of [16].*

On the other hand, I4 is generally not satisfied[9] and I7 is never satisfied. Thus, Lehmann is right when saying that for a belief change operator on a changing situation, "one would probably accept all the postulates [...] except I4 and I7" [16]. The difference between extrapolation and iterated revision is clear when considering the following example: let $\Sigma = \langle a \to b, a, \neg a\rangle$; any "reasonable" extrapolation operator satisfies $\Sigma\ddagger = \langle a \wedge b, a \wedge b, \neg a \wedge b\rangle$ (the change from a to $\neg a$ between 2 and 3 being certain, the preferred trajectory is the one containing no other changes). Now, Lehmann's iterated revision, and also most iterated revision operators defined on epistemic states (*e.g.*, [7] [2]) give $[a \to b, a, \neg a] = \neg a$. The reason for this difference is that iterated revision is concerned with pieces of information concerning a *static world*, therefore, once the new information $\neg a$ has "cancelled" the preceding one, the reasons to believe in b have disappeared. This strong "directivity" of time in iterated revision contrasts with extrapolation, where past and future can often be interchanged (as soon as the property (R) is satisfied).

5 EXTRAPOLATION AND UPDATE

An update operator is a function which associates to a knowledge base K representing knowledge about a system in an initial state and a new piece of information φ (whose precise meaning is to be discussed further) a new knowledge base $K \diamond \varphi$ representing the system after this evolution. As for revision, Katsuno and Mendelzon have proposed a set of postulates that an update operator $\diamond$ should satisfy. The most "typical" postulate is

U8 $(K1 \vee K2) \diamond \varphi \leftrightarrow (K1 \diamond \varphi) \vee (K2 \diamond \varphi)$.

which tells that models of K are updated indepen-

[8] Other approaches to iterated belief revision act on epistemic states rather than on sequences of formulas and the comparison with extrapolation is more complex to establish.

[9] It is satisfied if and only if $\preceq$ satisfies the property: for any $\Sigma = \langle \varphi_1, ...\varphi_N\rangle$ and any $\tau = \langle \tau(1)..\tau(N)\rangle$, $\tau \in Min(\preceq, Traj(\Sigma))$ implies $\langle \tau(1)..\tau(N-1)\rangle \in Min(\preceq, Traj(\langle \varphi_1, ..., \varphi_{N-1}\rangle))$. Only chronological operators such as $\ddagger_{chr}$ satisfy this property.

dently.

5.1 EXTRAPOLATION IS NOT BELIEF UPDATE

When $N = 2$, belief extrapolation projected on time 2 can be rephrased this way: from a belief set $K = \Sigma(1)$ at time 1 and an observation $\alpha = \Sigma(2)$ at time 2, compute a completed belief set at time 2. This may look similar to belief update, however 2-extrapolation *is not* belief update. This is clear from point 1 of Proposition 8 (or from E5): it is known that the AGM postulates are inconsistent with postulate U8 [12] as soon as the language contains at least two propositional symbols[10]:

Proposition 11 *Assume that the language contains at least two propositional symbols. Let $N = 2, \Updownarrow$ a 2-extrapolation operator, and $\star$ the induced belief change operator as in Section 4. Then $\star$ cannot satisfy U8. Therefore, no belief change operator induced by a 2-extrapolation operator is a belief update operator.*

This simple result has significant consequences. It means that for reasoning about time-stamped observations on a changing world, belief update is not adequate (see the example in Introduction). The key point is postulate U8 which, by requiring that all models of the initial belief set be updated separately, forbids us inferring new beliefs about the past from later observations. This inadequacy of update to handle observations is not new (see e.g. [4]), but is rarely mentioned in papers about belief update: indeed, saying that updating KB by α comes down to incorporate observation α (coming from sensors for instance) reflecting a change in the world is not correct.

So, the impossibility (with update) to infer new beliefs at time 1 from the incorporated belief at time 2 means that we do not *observe* (from sensors, by communicating with other agents etc.) anything at time 2[11]. Therefore, if not an observation, the input α is necessarily a *prediction*, more precisely, the projection of the expected effects of some "explicit change", most likely, the *expected* (not the *observed*) effect of an action.

[10] More precisely, U8 cannot be satisfied together with the strong K$\star$3 and K$\star$4 in presence of the more harmless postulates K$\star$2, K$\star$5 and K$\star$6.

[11] However, a second interpretation would be acceptable: there are observations at time 2 but they are completely unrelated to the world at time 1 (which implies giving up persistence assumptions and minimal change) and a belief change operator in this extreme case is not worth studying.

We see that the crucial issues are *observability* (what do we observe about the world at what time?) and *predictability of change*. Belief extrapolation deals with observation and unexpected change, while belief update is suitable for expected change without observation. In Sandewall's taxonomy [19], extrapolation is adequate for the action-free subclass of K-IS (correct knowledge, inertia and surprises) while update is adequate for the class K_p-IA (no observations after the initial time, inertia and alternative results of actions).

5.2 INTEGRATING EXTRAPOLATION AND UPDATE

Extrapolation and update complete each other and in order to be able to reason both with implicit and explicit change, we can integrate both rather easily as follows. A *mixed scenario* now contains both observations and explicit updates (corresponding to expected effects of actions) and is therefore a sequence of formulas labelled either by `observe` or by `update`, the first member of the sequence being of the `observe` type. For instance, $\Sigma_0 = \langle$ `observe:` $a \vee b$, `update:` $\neg b$, `observe:` $\neg a \rangle$ is a mixed scenario expressing that $a \vee b$ was first observed, then an action was performed whose expected effect is $\neg b$, and, after it, $\neg a$ was observed.

Let $\diamond$ be an arbitrary update operator and $\preceq$ a change-based preference relation induced by a relation $\preceq^c$ on change sets. Let Σ be a mixed scenario of length N and $\tau = \langle M_0, ..., M_N \rangle$ a trajectory. We say that τ satisfies Σ with respect to $\diamond$ if and only if (1) for every t such that $\Sigma(t) =$ `observe:`α we have $M_t \models \alpha$; (2) for every t such that $\Sigma(t) =$ `update:`α we have $M_t \in M_{t-1} \diamond \alpha$. Let $Traj_\diamond(\Sigma)$ the set of trajectories satisfying Σ w.r.t $\diamond$. Let $UnprCh(\tau)$ the set of unpredicted changes of τ, i.e., the changes $\langle f, t \rangle \in Ch(\tau)$ such that f is not involved in an update at time t. τ is a *preferred trajectory* with respect to $\diamond$ and $\preceq$ if and only if there is no trajectory $\tau' \in Traj_\diamond(\Sigma)$ such that $UnprCh(\tau') \prec^c UnprCh(\tau')$. Lastly, the operator $\Updownarrow$ induced by $\preceq$ and $\diamond$ is defined by $Mod(\Sigma \Updownarrow (t)) = \{\tau(t) \mid \tau \in Pref(Traj_\diamond(\Sigma))\}$. Therefore, preferred trajectories are selected in two steps: (1) filter out trajectories that do not satisfy all observations or all explicit changes; (2) among remaining trajectories, minimize unexpected change. Choosing any $\diamond$ satisfying the KM postulates and any $\preceq$, the latter Σ_0 yields $\Sigma_0 \Updownarrow = \langle a \vee b, \neg b, \neg a \wedge \neg b \rangle$.

This integration of extrapolation and update can deal with observations and actions (deterministic or not). Three issues are not dealt with:

1. *unreliable observations*, or misperceptions. These could be handled by attaching entrenchment levels to observations and then apply a temporal generalization of knowledge transmutations [21]. See also [5] and [6] for alternative ways of handling unreliable observations.

2. *normal and exceptional effects of actions*;

3. *complex events, whose expected effects yield an update, but whose occurrence is not known a priori but (abductively) inferred from the observations.* This issue is handled (at least for sequences of length 2) by Boutilier's *generalized update* [4].

All three issues could be dealt with by attaching degrees of surprise to misperceptions, event occurrences, action effects and event effects, and and then maximize the global level of surprise over the whole trajectory. This is a topic for further research.

6 RELATED WORK

In addition to the position of extrapolation with respect to revision, iterated revision, update and generalized update, already discussed in Sections 4 and 5, several works are related to belief extrapolation.

6.1 BERGER, LEHMANN AND SCHLECHTA

Berger et al. [3] propose a generic class of belief change operators which is actually a subclass of the set of extrapolation operators. More exactly, an extrapolation operator is a belief change operator as in [3] if and only if its preference relation satisfies the following property (BLS). Let $h(\tau)$ defined from τ by $h(\tau)(1) = \tau(1)$ and for $k \geq 1$, $h(\tau)(k+1) = \tau(i_{k+1})$ where i_{k+1} is the smallest integer such that $\tau(i_{k+1}) \neq \tau(i_k)$, if it exists; and so on until such an integer can no longer be found. The property (BLS) is then the following: for any trajectories τ, τ' such that there exists a sequence of integers $1 \leq j_1 < j_2 < ... < j_k$ such that $h(\tau) = \langle m_0, ..., m_n \rangle$ and $h(\tau') = \langle m_{j_0}, ..., m_{j_k} \rangle$, we have $\tau \preceq \tau'$. It can be shown that, among the preference relations we proposed, those that satifsy (BLS) are $\preceq_{nc}$, $\preceq_{ncf}$, $\preceq_{ifc}$, $\preceq_{cseq}$ and $\preceq^k_{cp}$. While they do not study various preference relations on trajectories nor investigate computational issues, Berger et al. get a both-directions representation result; we have not got such a result yet for belief extrapolation (note that our operators being less constrained than theirs, the latter representation result does not apply to belief extrapolation).

6.2 LIBERATORE AND SCHAERF

Liberatore and Schaerf [17] propose a fairly general system (BReLS) aiming at integrating revision, update and merging. It deals with time-stamped observations and consider two semantics; using the "trajectory" semantics and assuming that there is no more than one observation at each time point, we obtain our extrapolation operator induced by the penalty-induced relation $\preceq_{cp}$. The other semantics ("pointwise") yields iterated update (but is incompatible with extrapolation because of U8 which underlies this semantics); it seems that the authors did not consider mixing updates and observations by using both semantics simultanesouly as we briefly did in Section 5. On the other hand, BReLS provides an integrated treatment of both static and dynamic reasoning, and thus includes the possibility of reprsenting exceptional effects or misperceptions.

6.3 OTHER WORKS

Friedman and Halpern [11] define a very general framework for belief change, based on trajectories (or "runs") and in which the central process is that of *conditioning*. Like belief revision and update, it can be shown that extrapolation is an instance of their general framework.

Largouët et Cordier [15] propose a (probabilistic) framework for belief change integrating observations and expectations allowing, like ours, to complete an initial scenario by persistence assumptions. Their completion process operates in two subsequent steps: beliefs are first propagated forwards (from time 0 to time n) and then propagated backwards (from time N to time 0). Their process, by performing postdiction steps *after* prediction steps, give priority to backwards persistence and therefore almost never gives the same results as belief extrapolation.

In an earlier work [8] we considered a probabilistic model of unpredicted change which may be seen as the probabilistic counterpart of the approach given here. A prior probability is attached to each elementary change and assuming prior independence between occurrences of elementary changes. The case where probabilities of change are infinitely small was particularly focused on.

7 CONCLUSION

We have proposed a general family of belief change operators which consist in completing initial observations by persistence assumptions. We have discussed in de-

tails the links between extrapolation and revision, and its differences with belief update (and located both families of operators within Sandewall's taxonomy).

Further directions of work include :

- applying belief extrapolation to temporal diagnosis;
- developing a full framework for reasoning about both predicted change (action effects) and unpredicted change;
- generalize extrapolation to partially non-inertial systems where some fluents have a specific dynamics.

Acknowledgements

We would like to thank two anonymous reviewers for their useful comments, as well as Marie-Odile Cordier, Andreas Herzig, Sébastien Konieczny, Pierre Marquis and Odile Papini for helpful discussions.

References

[1] C. 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] S. Benferhat, S. Konieczny, O. Papini, and R. Pino-Pérez. Iterated revision by epistemic states: axioms, semantics and syntax. In *Proceedings of ECAI-2000*, pages 13–17, 2000.

[3] S. Berger, D. Lehmann, and K. Schlechta. Preferred history semantics for iterated updates. *Journal of Logic and Computatation*, 9(6):817–833, 1999.

[4] C. Boutilier. A unified model of qualitative belief change: a dynamical systems perspective. *Artificial Intelligence*, 1-2:281–316, 1998.

[5] C. Boutilier, N. Friedman, and J. Halpern. Belief revision with unreliable observations. In *Proceedings of AAAI-98*, pages 127–134, 1998.

[6] S. Coradeschi. Reasoning with misperception in the features and fluents framework. In *Proceedings of ECAI-96*, pages 657–661, 1996.

[7] A. Darwiche and J. Pearl. On the logic of iterated revision. *Artificial Intelligence*, 89:1–29, 1997.

[8] F. Dupin de Saint-Cyr and J. Lang. Reasoning about unpredicted change and explicit time. In *Lecture Notes in Artificial Intelligence (Proc. of ECSQARU-97)*, 1997.

[9] Th. Eiter and G. Gottlob. The complexity of logic-based abduction. *Journal of the Association for Computing Machinery*, 42(1):3–42, 1995.

[10] N. Friedman and J.Y. Halpern. Belief revision: A critique. In *Proc. of the 6^{th} KR*, pages 421–431, 1996.

[11] N. Friedman and J.Y. Halpern. Modelling beliefs in dynamic systems. part ii: revision and update. *JAIR*, 10:117–167, 1999.

[12] H. Katsuno and A.O. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proc. of KR'91*, pages 387–394, 1991.

[13] H. Katsuno and A.O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52:263–294, 1991.

[14] J. Lang and P. Marquis. Complexity results for independence and definability in propositional logic. In *Proc. of KR'98*, pages 356–367, 1998.

[15] C. Largouët and M.O. Cordier. Combining observations and expectations: application to the refinement of an image sequence classification. In *UAI workshop on Fusion of domain knowledge with data for decision support*, Standford, CA, USA, 2000.

[16] D. Lehmann. Belief revision, revised. In *Proc. of IJCAI'95*, pages 1534–1540, 1995.

[17] P. Liberatore and M. Schaerf. Brels: A system for the integration of knowledge bases. In *Proc. of KR'00*, pages 145–152, 2000.

[18] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[19] E. Sandewall. *Features and Fluents*. Oxford University Press, 1995.

[20] Y. Shoham. *Reasoning about Change - Time and Causation from the Standpoint of Artificial Intelligence*. MIT Press, 1988.

[21] Mary-Anne Williams. Transmutations of knowledge systems. In *Proceedings of KR'94*, pages 619–629, 1994.

Constraint Satisfaction on Infinite Domains: Composing Domains and Decomposing Constraints

Mathias Broxvall*
Department of Computer and Information Science
Linköpings Universitet
S-581 83 Linköping, Sweden
matbr@ida.liu.se

Abstract

We investigate the constraint satisfiability problem for infinite domains and demonstrate how tractable disjunctive constraints from disjunct domains can be combined to yield new tractable sets of relations. Inspired by these results, we also present a method that can be used to decompose problems into subproblems which can be solved independently. This method can be used to enhance any backtracking based constraint solver and we test the efficiency of this method on different ensembles of realistic problem instances.

1 Introduction

Many problems in computer science and elsewhere can be naturally expressed as constraint satisfaction problems. Although the constraint satisfaction problem is NP-hard in general, many special cases that can be solved in polynomial time have been identified [12] and much effort has been put into discovering new tractable cases and methods to construct new tractable cases from old ones [6]. In this paper we investigate the constraint satisfaction problem for primarily infinite domains such as Allen's interval algebra [1].

Previously, a method for dealing with disjunctions by identifying the independence property has been investigated [7]. We demonstrate that the independence property is preserved when building constraints from disjunct domains using the multiple relational union [6]. For many tractable constraint satisfaction problems, satisfiability is determined by path consistency. (For instance, all previously identified tractable cases of RCC-8 can be determined by path consistency [5].) We demonstrate that decidability by path consistency is preserved under the multiple relational union operator.

Inspired by the methods of ensuring tractability by restricting the allowed constraint relations, we also examine structural properties that can be exploited to solve problem instances more efficiently. More specifically, we identify a property among relations which can be used to decompose problem instances on a structural level to simpler problems which can be solved independently. Many other structural decomposition methods have been examined previously [9, 10] and although they all guarantee tractability of problem instances of a certain structure their major weakness is that they fail when used on problem instances which are not decomposable and where backtracking is necessary. Although the method we suggest does not ensure tractability other than for very specialized structures, it can be used to enhance the efficiency of almost any backtracking based solver for constraint satisfaction problems over infinite domains. We also propose an automatic method for checking this property. Furthermore, we test this method on realistic problems and note that it can yield significantly improved performance for certain domains and has a very small overhead cost otherwise.

We will continue this paper with a brief introduction to the constraint satisfaction problem. Section 3 contains new methods for identifying cases where problem instances over multiple domains can be solved in polynomial time. In Section 4 we give a decomposition method inspired by our earlier results and study it for a couple of example domains. This is followed in the last section by some tests on the practical speedup gained by utilizing this decomposition method on realistic problems.

*This research has been supported by the ECSEL graduate student program.

2 Preliminaries

We begin by recalling the general constraint satisfaction problem CSPSAT. The consistency problem CSPSAT($\mathcal{S}$) for sets $\mathcal{S} = \{R_i | R_i \subseteq D^{n_i}\}$ over a domain D is defined as follows:

Instance: A set V of variables over a domain D and a finite set Θ of constraints $\langle R, x_1, \ldots, x_n \rangle$, where $R \in \mathcal{S}$ is a constraint of arity n.

Question: Is there a total function $f : V \to D$ such that for each constraint $\langle R, x_1, \ldots, x_n \rangle \in \Theta$ the following holds: $\langle f(x_1), \ldots, f(x_n) \rangle \in R$.

The general CSPSAT problem is NP-hard, but many sets of relations have been found such that deciding satisfiability of CSPSAT problem instances containing only those relations are tractable. By using the multiple relational union of tractable sets of relations in different domains, Cohen *et al.* [6] have constructed new tractable sets of relations. Sometimes we refer to the constraint graph of a given problem instance. This is the graph consisting of a node for each variable in the instance and an edge (or hyper-edge) for each constraint connecting the variables in the constraint.

Definition 1 Let Γ_1 and Γ_2 be sets of relations over the non-empty sets D_1 and D_2 respectively. Then the *multiple relational union* of Γ_1 and Γ_2, written as $\Gamma_1 \dot{\bowtie} \Gamma_2$, is the following set of relations over $D_1 \cup D_2$:

$$\Gamma_1 \dot{\bowtie} \Gamma_2 = \left\{ R_1 \cup R_2 : \begin{array}{c} R_1 \in \Gamma_1, R_2 \in \Gamma_2 \text{ and} \\ \text{arity}(R_1) = \text{arity}(R_2) \end{array} \right\}$$

We also define two projection operators ρ_1, ρ_2 taking a composite relation $r \in \Gamma_1 \dot{\bowtie} \Gamma_2$ to its parts as follows:

$$\rho_1(r) = r \cap D_1^{\text{arity}(r)}$$

$$\rho_2(r) = r \cap D_2^{\text{arity}(r)}$$

These operators can also be applied to problem instances $\Pi = \langle V, C \rangle$ as follows:

$$\rho_1(\Pi) = \langle V, \{\langle \rho_1(R), x, y \rangle | \langle R, x, y \rangle \in C\} \rangle$$

$$\rho_2(\Pi) = \langle V, \{\langle \rho_2(R), x, y \rangle | \langle R, x, y \rangle \in C\} \rangle$$

In order to express disjunctive constraints we need to define the disjunction of relations:

Definition 2 Let R_1, R_2 be relations of arity i, j and define the disjunction $R_1 \vee R_2$ of arity $i + j$ as follows:

$$R_1 \vee R_2 = \left\{ (x_1, \ldots, x_{i+j}) \in D^{i+j} \;\middle|\; \begin{array}{c} (x_1, \ldots, x_i) \in R_1 \vee \\ (x_{i+1}, \ldots, x_{i+j}) \in R_2 \end{array} \right\}$$

Thus, the disjunction of two relations with arity i, j is the relation with arity $i + j$ satisfying either of the two relations. The disjunction of two sets of relation $\Gamma_1 \dot{\vee} \Gamma_2$ is the set of disjunctions of each pair of relations in Γ_1, Γ_2 plus the sets Γ_1, Γ_2. Furthermore: for any set of relations, Δ, define $\Delta^* = \bigcup_{i=0}^{\infty} \Delta^i$ where $\Delta^0 = \{\bot\}$ and $\Delta^{i+1} = \Delta^i \dot{\vee} \Delta$. The empty relation which is always unsatisfied is here written as $\bot$.

To prove tractability of disjunctive CSP languages, the independence property [7] is a useful tool.

Definition 3 For any sets of relations Γ and Δ, we say that Δ is *independent* with respect to Γ if for any set of constraints C in CSPSAT($\Gamma \cup \Delta$), C has a solution whenever every $C' \subseteq C$, which contains at most one constraint belonging to Δ, has a solution.

By identifying this property we can now solve certain classes of disjunctive constraints in polynomial time by using the following result given by Cohen *et al.* [7].

Theorem 4 For any sets of relations Γ and Δ, if CSPSAT($\Gamma \cup \Delta$) is tractable and Δ is independent with respect to Γ, then CSPSAT($\Gamma \dot{\vee} \Delta^*$) is tractable.

We will demonstrate how the multiple relational union and independence can be combined, giving us the possibility to reason tractably over problem instances with constraints from different domains and containing certain types of disjunctions. The following domains will be used as examples in this paper:

- Allen's interval algebra where the variables represent intervals on the real line. See [1] for definitions.
- The point algebra for totally ordered time where we reason about totally ordered points and have the primitive relations $<, =$ and $>$ defined in the obvious way. See also [4].
- The previous algebra can be extended to the point algebra for partially ordered time in which we allow the points to be partially ordered. The primitive relation then also contains the relation *parallel* ($\|$) which holds iff two time points are unordered. See also [4].
- Another point algebra for nonlinear time is the point algebra for branching time in which the points are ordered in a tree structure. It contains the relations $<, =, >$ and $\|$ as described above. For further information see [3].
- The spatial algebras RCC-5 and RCC-8. A good description can be found in Renz and Nebel [14].

3 Composing domains

In this section we will study the complexity of the constraint satisfaction problem over multiple domains. We show that decidability by path consistency is preserved under the multiple relational union. Furthermore, we demonstrate that in order to be able to handle disjunctive constraints from disjoint domains we need to restrict ourselves to *well-typed* problem instances, that is, problem instances where the domain of variables is unambiguously determined by the constraints. For instance, the problem instance containing only the following constraints from RCC-8 and the algebra for totally ordered time is not well-typed since it is not possible to derive whether z, w, k represents regions (belong to the domain of RCC-8) or time points.

$$x < y \vee z \text{ NEQ } w$$

$$z > \text{NEQ } k$$

We write the multiple relational union of the time-point relation $>$ and the RCC-8 relation NEQ as $>$ NEQ here. Formally, restricting our instances to be well-typed can be described as a requirement that for each variable v in a problem instance over domains $D_1, \ldots, D_n$ the constraint $\langle =_{D_i}, v, v\rangle$ should be included in the constraints where $=_{D_i}$ is the equality relation over some domain D_i. Note that the relation $=_{D_i}$ is a much more restrictive constraint than the general equality constraint $=$ which does not say anything about the domain of the variables. By only considering well-typed problem instances, we can solve certain disjunctive constraints tractably by noting that the independence property is preserved under the multiple relational union. We also demonstrate that without this restriction, the satisfiability problem is NP-hard as soon as even the simplest form of disjunctions are allowed. We begin with a quick note on how the composition operator behaves for relations over disjunctive domains.

Lemma 5 $(\gamma_1 \cup \gamma_2) \circ (\gamma_1' \cup \gamma_2') = (\gamma_1 \circ \gamma_1') \cup (\gamma_2 \circ \gamma_2')$ where γ_1, γ_1' and γ_2, γ_2' are relations over two disjunctive domains D_1 and D_2.

Proof sketch: Note that all the variables involved in the constraints either are all assigned values from D_1 or all from D_2. If the variables are assigned values from D_i the composition evaluates to $\gamma_i \circ \gamma_i'$, to preserve the choice of domain both versions must be maintained, thus $(\gamma_1 \circ \gamma_1') \cup (\gamma_2 \circ \gamma_2')$ □

Theorem 6 Let Γ_1 and Γ_2 be two tractable sets of relations over disjoint domains D_1 and D_2 such that path consistency implies global consistency, then consistency for $\Gamma_1 \bowtie \Gamma_2$ is decided by path consistency.

Proof: Let $\Pi = \langle V, C\rangle$ be an arbitrary path consistent problem instance over the relations $\Gamma_1 \bowtie \Gamma_2$ and define Π_1, Π_2 as follows:

$$\Pi_1 = \langle V, \{\langle \rho_1(R), x, y\rangle | \langle R, x, y\rangle \in C \wedge \rho_1(R) \neq \emptyset\}\rangle$$

$$\Pi_2 = \langle V, \{\langle \rho_2(R), x, y\rangle | \langle R, x, y\rangle \in C \wedge \rho_2(R) \neq \emptyset\}\rangle$$

We begin by showing that Π_1 and Π_2 are path consistent. Consider each triple of constraints $x\ \rho_i(R_1)\ y, x\ \rho_i(R_2)\ z, z\ \rho_i(R_3)\ y$ in Π_i. Since Π is path consistent and by using Lemma 5 we see that:

$$\rho_1(R_1) \cup \rho_2(R_1) \subseteq (\rho_1(R_2) \cup \rho_2(R_2)) \circ (\rho_1(R_3) \cup \rho_2(R_3)) =$$

$$(\rho_1(R_2) \circ \rho_1(R_3)) \cup (\rho_2(R_2) \circ \rho_2(R_3))$$

By noting that the relations in the given domains are closed under composition we see that $\rho_1(R_1) \subseteq \rho_1(R_2) \circ \rho_1(R_3)$ and $\rho_2(R_1) \subseteq \rho_2(R_2) \circ \rho_2(R_3)$. Thus, each such triple of constraints is path consistent and hence; Π_1, Π_2 is consistent.

Next, we will show that Π is consistent. Let f_1, f_2 be models of Π_1 and Π_2 respectively. We construct an interpretation of Π as follows:

$$f(x) = \begin{cases} f_1(x) & \text{Exists } \langle R, x, y\rangle \text{ or } \langle R, y, x\rangle \\ & \text{in } C \text{ such that } \rho_1(R) \neq \emptyset \\ f_2(x) & \text{otherwise} \end{cases}$$

Now consider an arbitrary constraint $c = \langle R, x, y\rangle \in C$. We consider two cases; If $\rho_1(R) \neq \emptyset$ then c is satisfied by f_1 and hence by f. Otherwise, since $\rho_1(R) = \emptyset$ we have that for all constraints $c' = \langle R', x, \_\rangle$ and $c'' = \langle R'', \_, x\rangle$ holds that $\rho_1(R') = \rho_1(R'') = \emptyset$ because of the path consistency of Π. Hence, $f(x), f(y) \in D_2$ and c is satisfied by f_2. Thus, all constraint of Π is satisfied by f and consistency for $\Gamma_1 \bowtie \Gamma_2$ is decided by path consistency. □

By noting that decidability by path consistency is preserved by the multiple relational union, we are not only able to solve problem instances using standardized algorithms but also have a necessary precondition for employing automated methods to identify the independence property [5] and partitioning relations (Theorem 14).

Only allowing binary constraints is (in practice) a severe limitation when modeling problems. Theoretically it is of course possible to model problems involving non-binary relations by using binary constraints over more complex domains. The limitation of having

only binary constraints is especially apparent in the context of combining several domains since a problem instance constructed only by binary constraints can essentially be seen as a number of disjoint, simpler problems, where the solutions of different subproblem has no influence on each other. For instance, given a problem instance over domains $D_1, \ldots, D_n$ with k sets of variables $V_1, \ldots, V_k$ such that the constraints occur only within each V_i, satisfiability of the the problem is simply asking whether for every V_i there exists some D_j such that $\rho_j(V_i)$ is satisfiable. This limits the expressibility since constraints such as "event A occurred before event B or region X is not the same as region Y" cannot be modeled. We continue by demonstrating why we need to restrict our problem instances to be well-typed in order to be able to decide satisfiability for these kind of constraints in polynomial time. For this purpose we need the definition of the NP-complete[8] problem MONOTONE 3SAT:

Instance: Set U of variables, collection C of clauses over U such that each $c \in C$ has $|c| = 3$ and contains either only positive or negative literals.

Question: Is there a truth assignment for U satisfying all clauses in C?

We can now by means of a reduction from this problem prove that deciding satisfiability is an NP-hard problem when we allow disjunctions of the multiple relational union of relations from two disjoint domains. This means that in order to be able to decide satisfiability of problem instances involving disjunctions and multiple domains we must restrict ourselves to well-typed problem instances.

Theorem 7 Let γ_1, γ_2 be two binary relations over disjoint domains D_1, D_2 respectively. Then CSPSAT$((\{\gamma_1\} \bowtie \{\gamma_2\})^2)$ is NP-hard.

Proof: We make a reduction from the NP-complete problem MONOTONE 3SAT. Let $P = \langle U, C \rangle$ be an arbitrary MONOTONE 3SAT problem instance. We construct a problem instance Π of CSPSAT$((\{\gamma_1\} \bowtie \{\gamma_2\})^2)$ as follows and show that Π is satisfiable iff P is satisfiable:

1. For each variable $v \in U$ let x_v, y_v be two fresh variables and add the constraint:

$$x_v(\gamma_1 \cup \gamma_2)y_v$$

2. For each positive clause $c = \langle i, j, k \rangle \in C$ add two fresh variables t_c, s_c and the constraints:

$$x_i\gamma_1 y_i \vee t_c\gamma_1 s_c$$

$$x_j(\gamma_1 \cup \gamma_2)s_c \vee x_k(\gamma_1 \cup \gamma_2)s_c$$

and for each negative clause:

$$x_i\gamma_2 y_i \vee t_c\gamma_2 s_c$$

$$x_j(\gamma_1 \cup \gamma_2)s_c \vee x_k(\gamma_1 \cup \gamma_2)s_c$$

Next, we prove that Π is satisfiable iff P is satisfiable. For the if-direction let $\mathcal{I}$ be a satisfying truth assignment for P, we construct an interpretation f of Π as follows.

$$f(x_i) = \begin{cases} d_1 & \text{if } \mathcal{I}(i) \text{ is true} \\ d_2 & \text{otherwise} \end{cases}$$

$$f(y_i) = \begin{cases} d_1' & \text{if } \mathcal{I}(i) \text{ is true} \\ d_2' & \text{otherwise} \end{cases}$$

$$f(t_{c^+}) = \begin{cases} f(x_j) & \text{if } \mathcal{I}(i) \text{ is true} \\ d_1 & \text{otherwise} \end{cases}$$

$$f(s_{c^+}) = \begin{cases} f(y_j) & \text{if } \mathcal{I}(i) \text{ is true} \\ d_1' & \text{otherwise} \end{cases}$$

where d_i, d_i' are two arbitrary values in D_i such that $d_i\gamma_i d_i'$ and c^+ denotes arbitrary positive tuples $\langle i, j, k \rangle$. Assignment of negative tuples if handled analogously as positive tuples by f. It is easy to see that each constraint of Π is satisfied by f and, hence, f is a model of Π. Thus Π is satisfiable whenever P is satisfiable.

For the only-if direction, let f be a model of Π and $\mathcal{I}$ the truth assignment assigning a the value true iff $f(x_a) \in D_1$. Obviously, each clause $\langle i, j, k \rangle$ is satisfied by $\mathcal{I}$ since either $x_i \in D_i$ or $s \in D_i$. If $s \in D_i$ then either x_j or x_k is also in D_i. Thus we have proven that Π is satisfiable iff P is satisfiable and, CSPSAT$((\Gamma_1 \bowtie \Gamma_2)^2)$ is NP-complete. □

Next, we show that by restricting the allowed problem instances to be well-typed we can solve some disjunctive constraints tractably by using the independence property. For these kinds of problem instances it is pointless to use the $\bowtie$ operator to combine relations since each such constraint in a problem instance can immediately be refined to one of its projections. Hence, we rather use the $\cup$ operator on sets of relations from disjoint domains.

Theorem 8 Let Γ_1 and Γ_2 be two sets of relations over disjoint domains D_1 and D_2 and $\Delta_1 \subseteq \Gamma_1, \Delta_2 \subseteq \Gamma_2$ two sets of relations independent of Γ_1, Γ_2 respectively. Then, $\Delta_1 \cup \Delta_2$ is independent of $\Gamma_1 \cup \Gamma_2$, for all well-typed problem instances.

Proof: Let $\Pi = \langle V, C\rangle$ be a problem instance of CSPSAT($\Gamma_1 \cup \Gamma_2$), such that for every subset $C' \subseteq C$ containing at most one constraint from $\Delta_1 \cup \Delta_2$, $\langle V, C'\rangle$ is satisfiable.

By showing that Π is satisfiable we prove that $\Delta_1 \bowtie \Delta_2$ is independent of $\Gamma_1 \bowtie \Gamma_2$. First, assume that the constraint graph of Π is connected. This implies that all variables in Π has been defined to be of the domain D_i so we consider $\rho_i(\Pi)$ which is a subproblem of Π. We note that all subproblems of $\rho_i(\Pi)$ containing at most one relation from Δ_i is also a subproblem of some set of constraints C' as described initially. Hence, each such subproblem of $\rho_i(\Pi)$ is satisfiable and since Δ_i is independent of Γ_i we have $\rho_i(\Pi)$ satisfiable. Thus, Π is satisfiable whenever the constraint graph of Π is connected.

For the case when the constraint graph is disconnected we simply apply the same reasoning on each component of constraints. □

The applicability of this result is obvious when we consider problems involving more than one domain. For instance, since disequality is independent of the point relations and since the set $\Delta_{\hat{\mathcal{H}}_8}$ is independent of the tractable $\hat{\mathcal{H}}_8$ fragment of RCC-8 [5] we see that we can tractably solve problem instances over time points and regions containing disjunctions of both the point relation disequality and the $\Delta_{\hat{\mathcal{H}}_8}$ relations. The possibilities of combining different domains with previous independence and tractability results is thus very promising.

4 The decomposition method

In this section, we are concerned with situations where we cannot solve our problem instances in polynomial time and need methods to speedup the solving process. We introduce the notion of partitioning relations and show how these can be used to decompose complex problem instances into several smaller and simpler instances that can be solved independently.

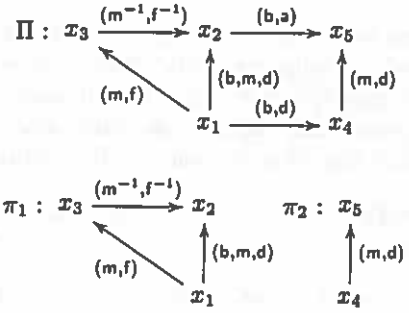
Definition 9 Let $\Pi = \langle V, C\rangle$ be a problem instance over Γ containing only binary constraints. A *partitioning* of Π is a problem instance $\pi = \langle\{\pi_1, \ldots, \pi_n\}, c\rangle$ with variables representing problem instances $\pi_1 = \langle v_1, c_1\rangle, \ldots, \pi_n = \langle v_n, c_n\rangle$ such that:

1. all $c_i \subseteq C$
2. $V = \bigcup_{i=1}^{n} v_i$ and $v_1, \ldots, v_n$ are disjoint.
3. For all binary constraints $x\ R\ y \in C$ holds $x\ R\ y \in c_i$ for some c_i or $x \in v_i, y \in v_j$ and $\exists R' \subseteq R : \pi_i\ R'\ \pi_j \in c$.
4. π is satisfiable.

It is here important to distinguish between relations and constraints. Since a relation is only a set of tuples from a domain there is no problem in condition three above to constraint the relation R' to be included in the relation R.

The concept of partitioning is further explained by an example below where we take a problem instance of five variables and split it into two partitions containing three and two variables respectively. The constraints within each of those partitions is preserved as they are. From the constraints between the two partitions we get a partitioning which is a problem instance of two variables π_1, π_2 and just one constraint which is trivially satisfiable.

Example 10 We can partition the Allen interval algebra problem instance Π below into the partitioning $\pi = \langle\{\pi_1, \pi_2\}, \{\pi_1\ (\mathsf{b})\ \pi_2\}\rangle$ with π_1, π_2 as given below. As we will see later this allows us to solve π_1, π_2 independently of each other.



Next, we present a property of relations that, in a way, resemble the independence property.

Definition 11 Let Γ be a set of binary relation over a domain D and $\varphi \subseteq \Gamma$ a set of relations such that: For every problem instance Π from Γ and every partitioning $\pi, \pi_1, \ldots, \pi_n$ of Π such that π contains only relations from φ; Π is satisfiable iff $\pi_1, \ldots, \pi_n$ is satisfiable. We say that φ *partitions* Γ.

By identifying relations which *partitions* other relations we can simplify the search problem of deciding satisfiability for problem instances by identifying partitionings consisting of these relations. Similar to the results in the previous section we show that the partitioning property of relations is preserved when combining domains.

Theorem 12 Let Γ_1, Γ_2 be sets of relations over the disjoint domains D_1, D_2. If φ_1, φ_2 is two subsets of Γ_1, Γ_2 such that φ_i partitions Γ_i then $\varphi_1 \cup \varphi_2$ partitions $\Gamma_1 \cup \Gamma_2$.

Proof: Let $\Pi = \langle V, C \rangle$ be a problem instance of CSPSAT($\Gamma_1 \cup \Gamma_2$) and $\pi, \pi_1, \ldots, \pi_n$ a satisfiable partitioning of Π. We prove that $\varphi_1 \cup \varphi_2$ partitions $\Gamma_1 \cup \Gamma_2$ by showing that Π is satisfiable iff $\pi_1, \ldots, \pi_n$ is satisfiable. The only if direction is trivial, for the if direction assume that $\pi_1, \ldots, \pi_n$ is satisfiable.

We can safely assume that the constraint graph for each partition π_i is connected. First, assume that the constraint graph of π is connected and that π contains only constraints over the domain D_i. It must then hold that each partition π_j contains only constraints over the domain D_i since the variables in π_j are connected to the variables in π. Then, each $\rho_i(\pi_j)$ and $\rho_i(\pi)$ is satisfiable and since φ_i partitions Γ_i we have that $\rho_i(\Pi)$ is satisfiable. Since the constraint graph for π and each π_j is connected we have that the constraint graph for Π is connected. This mean that $\Pi = \rho_i(\Pi)$ and Π is satisfiable.

When the constraint graph of π is not connected we can simply consider each component of the graph. □

As the next result demonstrates we can combine partitioning sets of relations easily. This gives us a more convenient description of sets of partitioning relations by only giving their minimal elements, since we implicitly know that their unions are also partitioning.

Theorem 13 Let φ be a set of relations partitioning Γ, then the set $\varphi' = \{\gamma_1 \cup \gamma_2 | \gamma_1, \gamma_2 \in \varphi\}$ partitions Γ.

Proof: Let Π be an arbitrary problem instance of Γ and $\pi', \pi'_1, \ldots, \pi'_n$ a satisfiable φ' partitioning of Π. Since π' is satisfiable it can be reduced to some $\pi \subset \pi'$ containing only relations from φ and hence, $\pi, \pi'_1, \ldots, \pi'_n$ is a satisfiable φ problem instance. Hence, Π is satisfiable iff $\pi'_1, \ldots, \pi'_n$ is satisfiable. Thus, we have proven that φ' partitions Γ. □

By using the following result we have an automatic method of deriving sets of partitioning relations.

Theorem 14 Let Γ be a set of relations over a domain D, $\psi \subseteq \Gamma$ such that consistency for ψ is determined by path-consistency and:

$$\forall r \in \Gamma : \exists r_1, \ldots, r_n \in \psi : r = r_1 \cup \cdots \cup r_n$$

For all sets of relations $\varphi \subset \psi$ we have that φ partitions Γ iff the constraints $\{x\ \gamma_1\ y, y\ \gamma_1^{-1}\ z, z\ \gamma_2\ x\}$ are consistent for all $\gamma_1 \in \varphi, \gamma_2 \in \psi$.

Proof: For the if-direction, let Π be an arbitrary problem instance of Γ and $\pi, \pi_1, \ldots, \pi_n$ a satisfiable partitioning of Π. We show that Π is satisfiable whenever $\pi_1, \ldots, \pi_n$ is satisfiable. Let $f, f_1, \ldots, f_n$ be a model of $\pi, \pi_1, \ldots, \pi_n$ respectively. We construct a new problem instance Π' containing all the variables of Π and the following constraints:

1. For each $x, y \in \pi_i$ add the constraint $x\ R\ y$ where R is the primitive relation containing $\langle f_i(x), f_i(y) \rangle$.
2. For each $x \in \pi_i, y \in \pi_j$ add the constraint $x\ R\ y$ where R is a relation in φ containing $\langle f(\pi_i), f(\pi_j) \rangle$.

Consider each triple of constraints $x\ R_1\ y, y\ R_2\ z, x\ R_3\ z$. We examine three cases. If x, y, z are all in one partition π_i, this triple of constraints must clearly be path consistent since they originate from the model f_i. Also, if they all belong to different partitions π_i, π_j, π_k they must also be path consistent since they are related only by the path consistent relations from f. For the remaining case when one of the variables comes from one partition and the two others from some other partition we see that $R_i \in \varphi, R_j = R_i^{-1}$ and $R_k \in \Gamma$ and hence, they are consistent. Thus, Π' is path consistent and since it only contains primitive relations, consistent. By noting that Π is a relaxation of Π' we see that Π is consistent and hence, the if-direction of the theorem holds.

Next, we prove the only-if direction. We show that the constraints $\Pi : x\ \gamma_1\ y, y\ \gamma_1^{-1}\ z, z\ \gamma_2\ x$ is consistent for all $\gamma_1 \in \varphi, \gamma_2 \in \Gamma$ by constructing a partitioning of Π assigning x, z to one partition and y to another. Obviously this partitioning is satisfiable and both partitions is satisfiable. Hence, Π is satisfiable. □

By using the previous theorem, it is now trivial to construct partitioning sets of relations for many relational algebras. We give only the primitive relations of the partitioning sets of relations here since inclusion of all their disjunctions follows from Theorem 13. The following sets of relations partitions the full set of relations for their respective domains.

- $\{<, >\}$ of the point algebra for linear time.
- $\{<, \|, >\}$ of the point algebra for partially ordered time.
- The relation $\|$ of the point algebra for branching time.

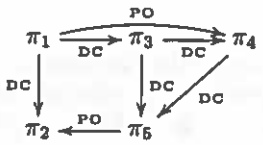
- $\{(b),(a)\}$ of Allen's interval algebra.
- $\{DC, PO\}$ of RCC-8 and $\{DR, PO\}$ of RCC-5.

Having identified partitioning relations for some domains we give now another example of partitioning a problem instance and see how this can be used to speed up the satisfiability calculations for this instance. Consider the RCC-8 problem instance Π given in Figure 1 which contains ten constraints not present in any (previously identified) tractable set of relations for RCC-8. If we where to solve this problem as a standard backtracking search problem [15] and non-deterministically split these constraints into constraints in some tractable set of relations we would in the worst case end up backtracking over a complete search tree of depth 10 containing several thousand nodes. However, by a simple preprocessing step we can partition the problem as follows:

$$\pi_1 = \{x_0, x_1, x_2\} \quad \pi_2 = \{x_6, x_7, x_8\}$$

$$\pi_3 = \{x_3\} \quad \pi_4 = \{x_4\} \quad \pi_5 = \{x_5\}$$

The constraints within each partition are left unchanged, and the constraints between partitions π are as given below:



Note that π is path-consistent and only contains relations present in $\hat{\mathcal{H}}_8$ which is a set of relations for which path-consistency decides consistency [14]. Also, all relations in π partitions the full set of relations for RCC-8. Hence, $\pi, \pi_1, \ldots, \pi_5$ is a satisfiable partitioning of Π. Thus, we can decide consistency of Π by checking the consistency for the partitions $\pi_1, \ldots, \pi_5$ independently of each other which can be done with very few nodes visited. Ideally this method yields a significant speedup for problem instances where a satisfiable partitioning can be found. In practice the original problem would be solved much more efficiently than visiting a complete search tree of depth 10 by employing better techniques to direct search and prune the search tree and therefore the actual speedup will not be as large as given in this example. However, as we will see in the test runs, the actual speedup is for most randomly generated problem instances enough to motivate the use of this partitioning method.

Next, we define partitionings on problem instances containing disjunctions and demonstrate how partitioning relations can be used there as well.

Definition 15 Let $\Pi = \langle V, C\rangle$ be a problem instance containing constraints from a set of binary relations Γ and disjunctions thereof. A *partitioning with disjunctions* of Π is a problem instance $\pi = \langle\{\pi_1, \ldots, \pi_n\}, c\rangle$ with variables representing problem instances $\pi_1 = \langle v_1, c_1\rangle, \ldots, \pi_n = \langle v_n, c_n\rangle$ such that:

1. all $c_i \subseteq C$
2. $V = \bigcup_{i=1}^{n} v_i$ and $v_1, \ldots, v_n$ are disjoint.
3. For all binary constraints $x\ R\ y$ in C or part of a disjunction in C holds $x\ R\ y \in c_i$ for some c_i or $x \in v_i, y \in v_j$ and $\exists R' \subseteq R : \pi_i\ R'\ \pi_j \in c$.
4. For each disjunction $d : x_1\ R_1\ y_1 \vee \ldots \vee x_m\ R_n\ y_m$ in Π it holds that either $x_i \in v_j, y_i \in v_k$ and $\exists R' \subset R_i : \pi_j\ R'\ \pi_k \in c$ for some i, j, k or $x_1, \ldots, x_m, y_1, \ldots, y_m \in v_i$ and $d \in c_i$ for some i.
5. π is satisfiable.

A *partial* partitioning (with disjunctions) is a partitioning with disjunctions not satisfying condition 4 above.

Example 16 If we add the constraint x_1 (f) $x_3 \vee x_1$ (b) x_5 to Π in Example 10 we get a partitioning with disjunctions. If we furthermore add the constraint x_1 (a, d) $x_2 \vee x_4$ (b, m^{-1}) x_5 to Π we only get a partial partitioning.

Our next result shows us that we can use partitioning relations to decompose problem instances containing disjunctions.

Theorem 17 Let Π be a problem instance containing binary and disjunctive constraints over Γ and let φ be a set of relations partitioning Γ. Furthermore, let $\pi, \pi_1, \ldots, \pi_n$ be a partitioning with disjunctions of Π such that π contains only relations from φ. Then Π is satisfiable iff $\pi_1, \ldots, \pi_n$ is satisfiable.

Proof: Proof by induction over the number of disjunctive constraints. The base case with zero disjunctions is trivial since $\pi, \pi_1, \ldots, \pi_n$ then is a satisfiable partitioning (without disjunctions) of Π. For the inductive step, let $d : x_1\ R_1\ y_1 \vee \ldots \vee x_m\ R_n\ y_m$ be a constraint of Π We have two cases: First assume that $x_i \in v_j, y_i \in v_k$ and $\exists R' \subset R_i : \pi_j\ R'\ \pi_k \in c$ holds for some i, j, k. If Π is satisfiable, then so is trivially $\pi, \pi_1, \ldots, \pi_n$. For the other direction, let Π' be Π with d refined as $x_i R_i y_i$, if Π' is satisfiable then so is Π. By noting that $\pi, \pi_1, \ldots, \pi_n$ is a partial partitioning of Π' also the induction hypothesis gives that Π' and Π is satisfiable. Hence, Π is satisfiable iff $\pi, \pi_1, \ldots, \pi_n$.

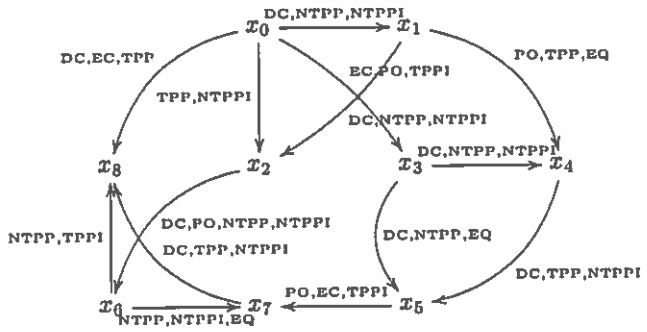


Figure 1: A problem instance for the domain RCC-8

Otherwise, we have $x_1, \ldots, x_m, y_1, \ldots, y_m \in v_i$ and $d \in c_i$ for some i. If some π_j is unsatisfiable then so is Π since π_j is a subproblem of Π. Otherwise, if each π_j is satisfiable we form π_i' as π_i with d substituted for one of its binary constraints $x_a\ R\ y_a$ which is satisfied by some model of π_i. Evidently π_i' is satisfiable. We also form Π' as Π with d substituted by $x_a\ R\ y_a$ and note that Π is satisfiable if Π' is satisfiable. Since $\pi, \pi_1, \ldots, \pi_{i-1}, \pi_i', \pi_{i+1}, \ldots, \pi_n$ is a satisfiable partitioning with disjunctions of Π' the induction hypothesis gives that Π' is satisfiable and hence, Π is satisfiable. □

Even in the case when we fail to identify a partitioning with disjunctions, we can use a partial partitioning to remove some of the (disjunctive) constraints in the problem and thus, making the problem easier to solve since less constraints need to be considered during backtracking.

Theorem 18 Let $\Pi = \langle V, C\rangle$ be a problem instance and $\pi = \langle v, c\rangle, \pi_1 = \langle v_1, c_1\rangle, \ldots, \pi_n = \langle v_n, c_n\rangle$ a partial partitioning of Π such that π only contains relations from a set of partitioning relations φ. Furthermore, form the set of disjunctions $M \subseteq C$ with a disjunct in different partitions:

$$M = \left\{ \begin{array}{c} d : x_{d_1} R_{d_1} y_{d_1} \vee \ldots x_{d_m} R_{d_m} y_{d_m} \in C \text{ such that} \\ \text{for some } d_i, d_j, d_k : x_{d_i} \in v_{d_j}, y_{d_i} \in v_{d_k} \text{ and} \\ \exists R' \subset R_{d_i} : \pi_{d_j}\ R'\ \pi_{d_k} \in c \wedge j \neq k \end{array} \right\}$$

and let Π' be Π with each constraint c such that $c \in M$ substituted for $x_{d_i} R_{d_i} y_{d_i}$. Then Π is satisfiable iff Π' is satisfiable.

Proof: If Π' is satisfiable then trivially so is Π. For the other direction, assume that Π is satisfiable. We form Π'' from Π by replacing each disjunctive constraint not in D by one of its binary constraints. Obviously, there exists at least one way of doing this so that Π'' is satisfiable. Next, note that $\pi, \pi_1, \ldots, \pi_n$ is a partitioning of Π'' and Theorem 17 gives that Π'' with each disjunction d in D replaced by $x_{d_i} R_{d_i} y_{d_i}$ is satisfiable. Since this is Π' with some disjunctions (those not in D) constrained to binary relations, we have that Π' is satisfiable. □

We describe a method for identifying a partitioning for arbitrary problem instances. Let φ be a partitioning set of relations and $\Pi = \langle \{v_1, \ldots, v_n\}, C\rangle$ a problem instance. We can now create a partitioning of Π as follows:

1. For each v_i create the partition $\pi_i = \langle \{v_i\}, \emptyset\rangle$. We denote the union of all relations in φ by $\top^\varphi$ and set $\pi = \langle \{x_1, \ldots, x_n\}, \{\langle \top^\varphi, x_i, y_j\rangle | 1 \leq i, j \leq n\}\rangle$
2. For each binary constraint $\langle R, x, y\rangle \in C$ and each disjunction containing such a constraint: Let R' be the relation such that $\langle R', x, y\rangle$ in π and refine this constraint to $\langle R' \cap R, x, y\rangle$.
3. Contract each pair of partitions π_i, π_j such that $\langle \emptyset, x_i, x_j\rangle \in \pi$. That is, replace each occurrence of x_j with x_i in π and set $\pi_i = \pi_i \cup \pi_j$. Calculate path consistency for π and repeat until π is path consistent.
4. Contract, in the same manner as in the previous step, two arbitrarily chosen partitions in π until π is consistent.
5. For each disjunctive constraint $d : x_1\ R_1\ y_1 \vee \ldots \vee x_m\ R_n\ y_m$ in Π such that $x_i \in v_j, y_i \in v_k$ and $\exists R' \subset R_i : \pi_j\ R'\ \pi_k \in c$ does not hold for any i, j, k contract all partitions containing one of $x_1, \ldots, x_m$ or $y_1, \ldots, y_m$.
6. For each constraint $c = \langle R, x_1, \ldots, x_n\rangle$ in C such that $x_1, \ldots, x_n \in \pi_i$ for some i, add c to the constraints of π_i.

no partitions	16	78	19	75	81	83	90	163	547	18046
with partitions	19	24	19	25	27	28	33	155	546	6028

Table 1: Number of nodes visited when solving a set of twelve problem instances for partially ordered time containing 40 variables and 150 disjunctions of binary constraints each.

Clearly, the algorithm above identifies only valid partitionings. In most cases φ is decided by path consistency and π will already be consistent in step four of the algorithm. In the cases where φ is not decided by path consistency, a domain dependent heuristic for choosing partitions to contract can be used to increase the chance of finding partitionings.

This method of identifying partitionings can be exploited by a solver in several ways. First, it can be used as a simple preprocessing step on instances containing only binary relations by splitting them into simpler instances and solving them independently. For problem instances containing disjunctions, the possibilities of identifying partitionings increase as disjunctions are replaced by binary relations during search. It is therefore worthwhile to continuously during search make new attempts to identify partitionings. Furthermore, the possibilities in which we can use the method is not limited to identifying partitionings with disjunctions and solve them independently. If we halt the algorithm before step five we get a partial partitioning which together with Theorem 18 gives us the opportunity of eliminating some of the constraints in order to speedup search.

5 Results

It is easy to construct problem instances which can be solved by the method described in the previous section thousands of times faster than when the method is not used. This can be done by for instance taking the disjoint union of several hard problem instances. In this section we test the efficiency of using this method on slightly more realistic problems by running it on different ensembles of randomly generated problem instances in different domains. For this purpose we have implemented a system that solves constraint satisfaction problems for arbitrary domains using backtracking and path-consistency algorithms. Our system also includes options to control the use of different heuristics such as forward checking and the partitioning method described here. The system as well as the problem instances on which it was tested is available for download from the authors web-site at:

`http://www.ida.liu.se/~matbr`

Many different methods of generating constraint graphs for empirically evaluating constraint solving algorithms have been suggested in the literature [2, 18, 11]. The perhaps simplest method of generating random problem instances is that of generating purely random constraint graphs. However, such graphs occur rarely in real life [17]. Rather than using purely random instances, constraint graphs with a *smallworld* [18] or *powerlaw* [2] structure has been suggested since they occur frequently in real world problems. We have therefore run our tests using both random, powerlaw and smallworld graphs.

We have run our system on both problem instances consisting purely of binary constraints as well as on problem instances containing disjunctions of binary constraints. Since the partitioning method can be combined with other heuristics such as forward checking and fail-first, there is a forward checking component in our system similar to that described by Stergiou and Koubarakis [16]. This is done by simply adding code that attempt to identify partitionings every nth time the forward-checking algorithm visits a new node in the search tree. If a partitioning is found then we solve the problem by recursively calling the forward checking algorithm on each partition.

A fairly typical example of the number of nodes visited when solving a set of problem instances can be found in Table 1. As can be seen from this example the time needed to solve problem instances vary greatly even though the problem size is kept constant. Thus, just a few of the problem instances will dominate the average of the solution time which would therefore give a poor measurement of the efficiency of the algorithms. Furthermore, simply comparing the median execution times of the algorithms is also a bad idea as this tells us nothing about the performance of the easiest and the hardest problems. We therefore choose to present our test data by sorting the problem instances by difficulty and give the average of the lower most, middle and topmost third. We also choose to present the number of nodes visited during search because it is easier to compare with other systems and it gives a more accurate and reproducible measurement compared to measuring runtimes. Comparing the total number of nodes visited when using the partitioning method and not gives a fair measurement since the execution time

Constraints	Part.	random			powerlaw			smallworld		
binary only	no	101	111	119	59	67	75	63	71	80
	yes	74	88	95	41	49	65	5.5	3.5	2.7
nearby disj.	no	83	120	76k	83	115	74k	115	1445	151k
	yes	34	92	72k	31	70	49k	46	1100	155k
random disj.	no	61	229	80k	87	184	58k	116	1064	124k
	yes	30	206	80k	36	125	37k	35	1378	116k

Table 2: Average number of nodes visited when solving problem instances from the point algebra for partially ordered time.

of computing partitionings is small compared to the total execution time.

We begin the investigations by comparing the efficiency of using our method by running tests on problem instances containing only binary relations. Furthermore, we have chosen to run the solver on only problem instances which are initially path consistent. We do this since problem instances that are not path consistent can rather be regarded as malformed problem instances than normal unsatisfiable problems since the cost of detecting and removing this problem instances is relatively small compared to the cost to solve problem instances for which we need to use backtracking. If we had chosen to not filter out the non path consistent problem instances the experiment would be completely dominated by the cost of performing the initial path consistency check since this would be enough to rule out most problem instances. Hence, the actual performance of the backtracking algorithms would not have been measured if we where using those problem instances.

Since we also want to test the efficiency of using the partitioning method on problem instances containing disjunctions, we need a method to generate such problem instances. The method we chose was to first generate a graph with either a random, powerlaw or smallworld structure just as for the binary problem instances. Edges from this graph was in turn picked in pairs to become the binary constraints in a disjunction of two binary constraints. We tested the difference between two different methods of picking these edges, either purely randomly or with a weighted probability where edges close to each other in the graph was more likely to be picked together.

We began by measuring the number of nodes visited when solving problem instances for the point algebra for partially ordered time. By varying the number of constraints and variables in the different ensembles we generated problem instances in the phase transition region. For the binary problem instances the constraint graphs contained 300 variables and had an average degree between 5 and 15 depending on the ensemble. For the disjunctive constraints we had between 25 and 50 variables and 175 to 220 constraints. As we will see, the gains of using the partitioning method on partially ordered time is sufficiently large for problem instances containing disjunctions to motivate its use. The success in this case can be attributed to the fact that as many as three of the four primitive relations can be present between partitions. Table 2 contains the average number of nodes visited when solving problem instances from these ensembles.

When testing the algorithms on Allen's interval algebra we choose problem ensembles containing between 40 and 100 variables and with an average constraint graph degree of 11 to 15 for the binary problem instances. For the problem instances containing disjunctions we had between 12 and 15 variables and 95 to 120 constraints. Using only between 40 and 100 variables for the binary problem instances may sound little. However, solving Allen algebra problem instances in the phase transition region is a hard computational problem, see eg. the experimental results of Nebel [13], and analyzing the algorithm on larger problem instances would be unpractical.

As can be seen in the results, the gains of using the partitioning method on this domain is not as large as for other domains. This can best be understood by noting that only two primitive relations, *before* and *after*, of the thirteen possible primitive relations can be present between partitions in problem instances for this algebra. The likelihood of partitions of relations occurring in randomly generated problem instances for this algebra is thus not as big as for the point algebra for partially ordered time and this is reflected in the number of visited nodes in Table 3. Note however that the poor results of this method on randomly generated problem instances does not necessarily imply that the method should not be used for Allen's interval algebra. Even though it does not yield much speedup for most problem instances it can give high gains for some highly structured problem instances and although it can give

Constraints	Part.	random			powerlaw			smallworld		
binary only	no	25	86	1754	30	95	838	4.6	9.6	694
	yes	24	82	1773	32	131	854	4.8	8.8	674
nearby disj.	no	610	5652	94k	48	1149	67k	1120	4582	118k
	yes	514	4621	90k	44	1138	66k	1002	4153	116k
random disj.	no	734	4889	90k	532	13k	158k	583	7120	78k
	yes	733	5080	113k	530	15k	141k	582	7099	78k

Table 3: Average number of nodes visited when solving problem instances from the Allen algebra.

a slight performance loss in some cases it more often than not gives a performance boost even on randomly generated problem instances.

We have also performed tests on several other domains such as the point algebra for linear time, RCC-5 and RCC-8. The improvements for these domains lies between that for Allen's interval algebra and that for the point algebra for partially ordered time. We note that the method is more successful for problem instances from domains with a larger proportion of partitioning relations and with more structure than random graphs.

It can reasonably be expected that the success of using the partitioning method on problem instances from different domains is dependent on the proportion of partitioning relations in that domain. For instance, for partially ordered time it is quite successful since 75% of the primitive relations partitions the rest of the relations. For the Allen algebra where only 15% of the primitive relations partitions the rest the results of the test runs is not as promising. If this holds in the general case it should be fairly efficient to use this method for the point algebra for linear time where 67% of the primitive relations can be used in partitions and less efficient but still useful for RCC-5 and RCC-8 where 40% and 25% respectively, of the relations can be used. Some test runs for these domains can be found in Table 4. Note that we do not provide data for binary problem instances for linear time since the full set of binary relations for linear time can be solved by path consistency so only one node need to be visited to solve these kind of problem instances.

6 Conclusions

In this paper we have examined the constraint satisfaction problem and given new methods of deriving tractable classes containing disjunctions for problem instances combining disjoint domains. We have investigated how to efficiently solve the constraint satisfaction problem for intractable cases and identified a property which enables us to split problem instances into smaller and simpler instances which can be solved independently. Furthermore, we also give an automatic method to test for this property. By implementing a general system for solving such problems, we have also been able to test the efficiency of using the methods described on realistic problem instances. As was expected the results vary greatly depending on the domain and the type of problem instances generated. The benefit of using this method is much greater on domains where a larger proportion of the relations can be used in partitionings.

References

[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.

[2] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[3] M. Broxvall. The point algebra for branching time revisited. In *Proceedings of the Joint German/Austrian Conference on Artificial Intelligence (KI-2001)*, Vienna, Austria, 2001.

[4] M. Broxvall and P. Jonsson. Disjunctive temporal reasoning in partially ordered time structures. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 464–469. AAAI Press, 2000.

[5] M. Broxvall, P. Jonsson, and J. Renz. Refinements and independence: A simple method for identifying tractable disjunctive constraints. In *Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming*, pages 114–127, 2000.

[6] D. Cohen, P. Jeavons, and R. Gault. New tractable classes from old. In *Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming*, pages 160–171, 2000.

Disj.	Structure	Part.	lin			rcc-5			rcc-8		
none	random	no	1			156	171	179	292	398	2731
		yes	1			138	148	159	294	396	2731
	powerlaw	no	1			22	28	32	150	162	175
		yes	1			11	8.0	5.9	77	79	79
nearby	random	no	62	75	6986	60	222	62k	162	3403	97k
		yes	26	43	6616	43	236	64k	137	3376	74k
	powerlaw	no	38	71	3573	62	110	26k	54	109	8205
		yes	18	31	727	44	100	22k	43	95	8124
random	random	no	81	95	22k	61	94	20k	136	2020	89k
		yes	34	100	18k	41	80	21k	141	2284	90k
	powerlaw	no	60	97	13k	46	82	38k	74	1280	71k
		yes	22	59	7652	31	63	37k	67	1309	67k

Table 4: Avg. number of nodes visited for the point algebra for linear time, RCC-5 and RCC-8.

[7] D. Cohen, P. Jeavons, P. Jonsson, and M. Koubarakis. Building tractable disjunctive constraints. *J. ACM*, 47(5):826–853, 2000.

[8] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, New York, 1979.

[9] G. Gottlob, N. Leone, and F. Scarcello. A comparison of structural CSP decomposition methods. *Artificial Intelligence*, 124:243–282, 2001.

[10] M. Gyssens, P. G. Jeavons, and D. A. Cohen. Decomposing constraint satisfaction problems using database techniques. *Artificial Intelligence*, 66:57–89, 1994.

[11] T. Hogg. Refining the phase transition in combinatorial search. *Artificial Intelligence*, 81(1-2):127–154, 1996.

[12] A. Krokhin, P. Jeavons, and P. Jonsson. Reasoning about temporal relations: The tractable subalgebras of allen's interval algebra. Technical Report PRG-RR-02-12, Oxford University, 2001.

[13] B. Nebel. Solving hard qualitative temporal reasoning problems: Evaluating the efficiency of using the ORD-Horn class. *Constraints*, 1(3):175–190, 1997.

[14] 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.

[15] J. Renz and B. Nebel. Efficient methods for qualitative spatial reasoning. *Journal of Artificial Intelligence Research (JAIR)*, 15:289–318, 2001.

[16] K. Stergiou and M. Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence*, 120(1-2):81–117, 2000.

[17] T. Walsh. Search on high degree graphs. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-2001)*, pages 266–271, 2001.

[18] D. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

Judgments about spatio-temporal relations

Thomas Bittner
Qualitative Reasoning Group, Department of Computer Science, Northwestern University
and
Institute for Formal Ontology and Medical Information Science at the University of Leipzig

Abstract

We define a family of qualitative spatio-temporal relations such as same-place-same-time and same-path-different-time, which describe the relative location of spatio-temporal objects within places or along paths. The relations in question are approximate, and this means that some of them are context-dependent. We explore the relationships between context, judgments that are made in certain contexts, and the spatio-temporal relations that do occur in those judgments. Understanding these relationships is important for understanding human judgments about spatio-temporal configurations as well as for the interaction of humans with spatio-temporal databases.

1 Introduction

Spatio-temporal databases and other spatio-temporal knowledge representation systems are of increasing importance, in the light of the fact that database technology has improved in ways which, allow databases to take into account the inherently dynamic character of our surrounding world. Modern spatio-temporal database technology is or will soon be able to represent the fact that things move, change, appear, and disappear over time.

In this paper we focus on *spatio-temporal* relations, like same-place-same-time, same-place-different-time, or different-place-different-time. Unfortunately, notions like 'same place' and 'same time' are problematic, since their meaning depends on context. For example, in order to say that two electrons are in the same place at the same time our intuition tells us that both must overlap, at least momentarily in a way which can be achieved only by using a particle accelerator. On the other hand in order to say that John and Mary are in the same place at the same time it is sufficient, depending on the context, that they are kissing each other, or that they are holding hands, or that they are in the same room. In everyday contexts, being at the same place in the same time does not require spatio-temporal objects actually to overlap.

While people seem to deal effortlessly with the different meanings of same-place-same-time in varying contexts, it is hard to formalize the context dependence that is thereby involved. Imagine a database that continuously tracks the GPS-location of the cell-phones of two terrorists, X and Y, and imagine that an FBI-investigator tries to find out whether X and Y have met by querying the database. She could start by extracting whether or not the spatial distance between X and Y dropped below a certain threshold, say 10 meters. From this, however, it does not follow that X and Y have met, since they might have been at the locations in question at different times. Adding a threshold of temporal distance does not necessarily help: For example X and Y might have been sitting for three minutes in two distinct subway trains five meters apart without having had the chance to meet. They have been in *different* places at the same time and it does not matter how close their actual distance was.

No matter how problematic the context-dependence of spatio-temporal relations like same-place-same-time and different-place-same-time might be, such relations are among the most important targets of spatio-temporal queries. This paper presents an effort towards a formalization of such qualitative spatio-temporal relations.

2 Spatio-temporal objects and their location

2.1 Objects and their location

Spatio-temporal objects fall into two major categories: continuants and occurrents (Simons 1987). Continuants change over time, for example by gaining or loosing parts, by growing older, by changing their location, but yet remain the same thing. Examples are human beings like

John and Bill Clinton, cars, substances in general. Occurrents do not change over time. They just occur. Examples are 'John's flight to New York', 'World War 2', 'The life of Bill Clinton', 'My childhood', and so on. The existence of an occurrent always depends on the existence of some continuant. The occurrent 'John's childhood' cannot exist without the continuant John. We define the relation *continuant-of*(c, o), which holds between an occurrent o and the continuant c it depends on, e.g., *continuant-of*$(John, John's\ childhood)$.

The relationship between occurrents and continuants is very complex in the sense that a single occurrent may depend on multiple continuants and in the sense that the set of continuants it depends on can change, i.e., loose and gain members. We assume for simplification: (a) that there is a single continuant for each occurrent to depend upon; and (b) that this continuant does not disappear or gets replaced during the time-intervals under consideration. Under these (very strong) assumptions the relation *continuant-of* is a functional relation, i.e., $John =$ *continuant-of*$(John's\ childhood)$.

Considering the relationship between occurrents and the continuants they depend on we can distinguish two major categories of occurrents: events and processes. Events are spatio-temporal objects that correspond the occurrence of a certain continuant in a certain state, e.g., the occurrent 'John's childhood' corresponds to the continuant John in the state of being a child. Processes, on the other hand, correspond to the occurrence of the change of some continuant. For example, the process of 'John's growing up' corresponds to certain patterns of changes of the continuant John.

We distinguish the domain of spatio-temporal objects, O, and the domain of regions, R, which both are considered as sets. The domain of regions is constituted by regions of different dimensionality which satisfy the axioms of the RCC-theory (Randall, Cui & Cohn 1992): four-dimensional spatio-temporal regions x, three-dimensional spatial regions, x^s, and one-dimensional temporal regions, x^t. The relationship between individual objects and individual regions is established by the notion of location. We write $L(o, x)$ in order to denote that the object o is located at the region x [1].

Every (four-dimensional) spatio-temporal object, $o \in O$, is exactly located at a single three-dimensional spatial region, x^s, at every instant of time, τ (Casati & Varzi 1995): $\forall o \in O : \exists x^s \in R : L^s_\tau(o, x^s)$. The region x^s is the exact or precise spatial location of o at the time instant τ. Since every spatio-temporal object is exactly located at a single region of space at a particular instant of time the relation $L^s_\tau(o, x^s)$ is a total functional relation, $\rho_\tau : O \to R$ with $x^s = \rho_\tau(o) \equiv L^s_\tau(x^s, y)$. We extend the instant-based localization function, ρ_τ, by adding an additional parameter τ ranging over instants of time. The resulting localization function is of signature: $\rho : \mathcal{T} \times O \to R$, where $\mathcal{T}$ is the time-line with all its time-instants.

Every spatio-temporal object, o, is temporally located in a unique region of time, $x^t \in R$, bounded by the beginning and end of its existence. The temporal location is a functional relation.

If we assume that space and time are distinct kinds of dimensions then spatio-temporal location is a *ternary* relation between (1) spatio-temporal objects; (2) temporal regions; and (3) regions of space, i.e., $L(o, x^t, x^s)$.

2.2 Change of spatial location

There are continuants that have different exact spatial locations at different times. We say that these objects change their spatial location. If we consider a temporal region (a period of time) during which the spatio-temporal object o existed then (i) o may be *at rest*, i.e., it may be located in the same region of space over a period of time, or (ii) its spatial location may change, by being located in different regions of space at different time-instants during this period. An example for (i) is John's location described in 'John was *at home* this morning' and an example for (ii) is John's location described in 'John was *on his way home* this morning'.

Consequently, there are two ways of defining spatio-temporal location in terms of ternary relations and we use the notations $L_R(o, x^t, x^s)$ and $L_P(o, x^t, x^s)$ in order to refer to those relations. More precisely we define:

The relation $L_R(o, x^t, x^s)$ holds if and only if the spatio-temporal object o is spatially at *rest* at x^s over x^t, i.e., its spatial location is identical for all time-instants within x^t:

$$L_R(o, x^t, x^s) \equiv \forall \tau \in x^t : \rho(\tau, o) = x^s, \qquad (1)$$

where $\tau \in x^t$ abbreviates: the time-instant τ lies within the boundaries of the temporal region x^t. We assume that there is no state of rest that lasts only for a single time-instant.

The relation $L_P(o, x^t, x^s)$ holds if and only if o *changes within* x^s *during* x^t. The spatial region x^s is the mereological sum of all regions at which the object, o, was exactly

[1] We say that the object o is located *at* the (spatio-temporal) region x in order to stress the exact fit of object and region (the object matches the region). It is important to distinguish the exact match from the case of an object being located *in* a region where the region is allowed to be bigger than the object. Both are distinguished from the case of an object *covering* a region, which intuitively implies the region is smaller than the object.

spatially located over the time-period x^t:

$$L_P(o, x^t, x^s) \equiv x^s = \bigvee_{\rho(\tau,o)} \tau \in x^t. \quad (2)$$

3 Approximate spatio-temporal location

Often, however, it is not very interesting to know that John is exactly located at that region of space from which the air is displaced by his body at a particular instant in time. It is much more interesting to know, for example, that John is in London or in Hyde Park. Hence we define a family of functions $\lambda^1_\tau, \ldots, \lambda^n_\tau : O \to R$, each of which yields a unique region of space for each spatio-temporal object $o \in O$ at instant τ. For every λ^i_τ this region is such that the exact location of the object in question is a mereological part of it, i.e., $\rho_\tau(o) \leq \lambda^i_\tau(o)$. We can think of a particular $\lambda^i_\tau(o)$ as an approximate location of o at instant τ.

The definitions of exact spatio-temporal location, $L_R(o, x^t, x^s)$ and $L_P(o, x^t, x^s)$ given above above can be generalized easily to take into account the approximative character of spatial location, temporal location, or both. In order to formalize approximate spatial location we use approximate localization functions, λ^i, rather than exact localization functions ρ. Formally we define corresponding to Definition (1): $\mathcal{L}^i_R(o, x^t, x^s) \equiv \forall \tau \in x^t : \lambda^i(\tau, o) = x^s$. We say that x^s is the place (marking the approximate location) where the event or process o occurred over the time-period x^t (or the place in which the continuant o was at rest over x^t). This does justice to the intuition behind the approximate character of John's spatial location in the sentences 'John spent Monday morning in Buckingham Palace'. The superscript i indicates the dependence of $\mathcal{L}$ on the underlying approximation function λ^i.

Imagine that we intercept the communication of the Queen's Secret Services agents as they trace John's movement from the entrance through the hall to the guest-room. If we sum up all those reported regions (entrance + hall + ...+ guest-room), then we get John's approximate path of movement, x^s, during x^t, through the palace. Formally we define corresponding to Definition (2): $\mathcal{L}^i_P(o, x^t, x^s) \equiv x^s = \bigvee_{\lambda^i(\tau,o)} \tau \in x^t$.

Also, it is normally not very interesting to know that John was at a certain location for exactly 0.34768410^{10} nanoseconds. Instead of saying that John was in Hyde Park *exactly* from 10.00 a.m. to 11.00 a.m., people would often rather say that he was there on Monday morning. Even in the case where people say that John was in Hyde Park from 10.00 a.m., they do not mean that he crossed the boundary of the park at exactly 10.00 and not a second earlier or later. Consequently, if we want to do justice to ordinary reasoning, we need to define a notion of spatio-temporal location based on the approximate temporal extent of the rest or the change of spatio-temporal objects and define: [2]

$$\Lambda^i_R(o, x^t, x^s) \equiv \exists x \forall \tau \in x : \lambda^i(\tau, o) = x^s \wedge x \leq x^t \quad (3)$$

$$\Lambda^i_P(o, x^t, x^s) \equiv \exists x : x^s = \bigvee_{\lambda^i(\tau,o)} \tau \in x \wedge x \leq x^t \quad (4)$$

4 Context and approximate location

At each time-instant, τ, there is a single *context-free* exact localization function ρ_τ and arbitrarily many *context-dependent* approximate localization functions λ^i_τ. The exact localization function is context-free because it is same for all contexts. Approximate location is context-dependent because in a given context there is one and only one approximate localization function $\lambda^i_\tau(x)$. Those localization functions are used in order to make reference to spatio-temporal location in the given context.

The notion of context is difficult and a formalization goes beyond the scope of this paper. For us it is sufficient to assume that each context is characterized: (1) by a specific period of time over which it is active; (2) by the feature of selectivity limiting its scope to only certain objects; and (3) the feature of granularity determining the coarseness of reference to spatio-temporal location. Those aspects determine the properties of the approximate localization function associated with a given context.

4.1 Limited scope, selectivity, the notion of rest

In order to reflect the limited scope of context and associated approximation we extend instant-based approximate localization functions, λ^i_τ, by adding an additional parameter τ ranging over instants of time. In order to 'glue' approximation functions to their context we consider them to be partial functions with respect to their temporal parameter. This means that they are defined only while the corresponding context is active. *Context-dependent* approximate localization functions are of signature: $\lambda^i : T \times O \to R$ where $T \leq \mathcal{T}$ is the period of time during which the context i is active.

Another important feature of context is its selectivity. In a specific context we are not interested in the function λ^i in its range over all objects in the universe. When asking John about his whereabouts, we are not interested in the location of some statue in China. We are interested in the location

[2] One might want even weaker definitions in order to be able to capture also sentences of the form 'John was in Hyde Park from 10 to 11 a.m.', which intuitively would not be false in an everyday conversation, even if John entered the park at 9.55 or 10.05 a.m. and left it at 10.55 or 11.05.

of just some few selected objects that are in the *foreground* of our attention (John, Buckingham Palace, London, ...). Consequently we assume that in specific contexts the approximate localization functions, λ^i, are partial functions also in respect to the objects in their range.

Given the context-dependence of approximate location then the term 'rest' is meaningful only within a fixed context, i.e., with respect to the fixed approximate localization function associated with this context. Consider the approximate localization function λ^i which yields 'Buckingham Palace' for the object John and the time-period 'Monday morning' and the approximate localization function λ^j which yields 'London' regarding John's whereabouts on Monday morning. With respect to the approximate localization function λ^i being at rest means that John does not leave the palace. With respect to the approximate localization function λ^j being at rest means that John does not leave London.

4.2 Systems and levels of granularity

Consider the sentence 'John spent Monday at Buckingham Palace'. So far we have considered 'Buckingham Palace' as a the name of spatial region in which some spatio-temporal object is at rest. If we take a closer look then it turns out that 'Buckingham Palace' is the name of a *place*, which happens to be located at the same spatial region within which the continuant John is at rest or where the event 'John spending this Monday morning' occurs. In the remainder we will use the notion of place in order to refer to spatial locations where certain continuants are at rest and where certain events occur. Places are organized in a hierarchical fashion. We call those structures *systems of granularities*.

Granularities are the results of the way we humans structure our surrounding world and provide the foundation for the notion of approximation and for reasoning about approximations (Bettini, Wang & Jajodia 1998, Smith & Brogaard to appear, Bittner & Smith 2001*a*). In the context of approximation of spatial and temporal location the (singular) notion of granularity refers to the size of the approximating region. The plural notion of granularities then refers to hierarchically organized systems of regions. In our example above the regions referred to by the names 'Hyde Park' and 'London' belong to such a system of granularities.

Formally, a system of granularities is a pair, $G = (R, \subseteq)$, where R is a set of regions with a binary relation $\subseteq$. Following (Smith & Brogaard to appear) we call those regions cells and the relation $\subseteq$ the subcell relation. Systems of granularities are governed by the following axioms (Bittner & Smith 2001*a*): (G1) $\subseteq$ is reflexive, transitive, and antisymmetric; (G2) Systems of granularities have unique maximal cells, called 'the root-cell', i.e., $\exists g \in G : \forall g_1 \in G : g_1 \subseteq g$; (G3) each cell g in a system of granularities is connected to the root cell by a finite chain: i.e., $\exists g_1, \ldots, g_n : g \subseteq g_1 \subseteq \ldots \subseteq g_n \wedge root(g_n)$; (G4) Two cells overlap each other if and only if one is a subcell of the other, i.e., $\exists z : (z = z_1 \; o \; z_2) \rightarrow (z_1 \subseteq z_2 \vee z_2 \subset z_1)$. It follows that every system of granularities can be represented as a tree structure, i.e., as rooted directed graphs without circles (Bittner & Smith 2001*a*), by taking the regions as nodes and by demanding that there is an edge from node a to node b if and only if $a \subseteq b$.

Consider the following examples: (E1) A spatial system of granularities is formed by the cells Hyde Park, Soho, Buckingham Palace, Downtown, London, York, Edinburgh, Glasgow, England, Scotland, Great Britain, Germany, Europe and the corresponding nesting of those cells (Figure 1); (E2) The political subdivision of the United States forms a (flat) system of granularities with the US as root-cell and minimal cells like Wyoming and Montana; (E3) A temporal system of granularities is formed by the subdivision of Saturday, January 13th 2002 into forenoon, afternoon, hours, half-hours, quarters, and five-minute slots.

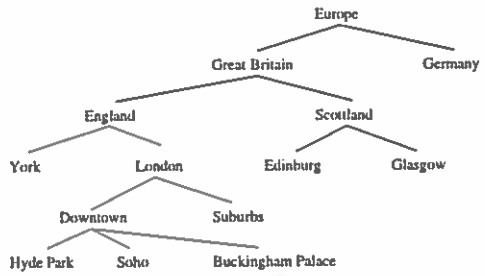


Figure 1: A system of granularities

A closer look at the examples E1-3 reveals that there are systems of granularity which are such that cells at one level sum up the next superordinate cell in the sense in which the Federal States sum up the US and in the sense in which three five-minute slots sum up quarter-of-an-hour slots etc. We call those systems of granularities *full* (Bittner & Smith 2001*b*). On the other hand there are systems of granularities like the one in E1 which do not have this property. Temporal systems of granularities are often full where systems of granularities formed by places usually lack the property of fullness.

Let $G = (R_G, \subseteq)$ be a system of granularities and let $\mathcal{G}$ the corresponding tree representation. A *level of granularity* in G then is a cut in the tree-structure in the sense of (Rigaux & Scholl 1995): (1) Let X be the root of $\mathcal{G}$, then $\{X\}$ is a cut; (2) $sons(X)$ is a cut, where $sons(a)$ is the set of immediate descendants of a; (3) Let C be a cut and $v \in C$ such that $sons(v) \neq \emptyset$ then $C' = (C - v) \cup sons(v)$ is a cut.

This definition ensures that (i) the elements forming a level of granularity are pair-wise disjoint, i.e., $\neg\exists v_1, v_2 \in C : v_1 \subset v_2 \vee v_2 \subset v_1$; (ii) levels of granularity are exhaustive in the sense that $\forall v \in R_G :$ if $v \notin C$ then $\exists v' \in C : v \subseteq v' \vee v' \subseteq v$.

Consider Figure 1. Levels of granularity, for example, are:

$$\begin{array}{ll} g_0 & \{Europe\} \\ g_1 & \{Great\ Britain, Germany\} \\ g_3 & \{York, London, Scotland, Germany\} \\ g_4 & \{York, Hyde\ Park, Soho, Buckingham\ Palace, \\ & \quad Suburbs, Edinburgh, Glasgow, Germany\} \\ \ldots & \end{array} \tag{5}$$

4.3 Context and granularity

Contexts have associated a unique localization functions of signature $\lambda^i : \mathcal{T} \times O \rightarrow R$ which are partial regarding their temporal parameter and partial regarding the objects in their range. We now argue that those context-dependent approximate localization functions target only spatial regions which belong to a *single* level of granularity which is fixed for the given context.

Let C be a context which is active over the time-period T_C and which has associated systems of spatial and temporal granularities $\mathcal{G}^s_C$ and $\mathcal{G}^t_C$, and let $\mathcal{L}_{\mathcal{G}^s_C}$ and $\mathcal{L}_{\mathcal{G}^t_C}$ be levels of granularities within these systems (with $\forall g \in \mathcal{L}_{\mathcal{G}^t_C} : g \leq T_C$ at the temporal level). A context-dependent approximation function is then of signature $\lambda^C : T_C \times O_C \rightarrow \mathcal{L}_{\mathcal{G}^s_C}$.

Context-sensitive ternary relations of approximate location are then defined as:

$$\begin{array}{l} \Lambda^C_R(o, x^t, x^s) \equiv x^t \in \mathcal{L}_{\mathcal{G}^t_C} \wedge \\ \quad \exists x : \forall \tau \in x : \lambda^C(\tau, o) = x^s \wedge x \leq x^t \end{array} \tag{6}$$

and

$$\begin{array}{l} \Lambda^C_P(o, x^t, x^s) \equiv x^t \in \mathcal{L}_{\mathcal{G}^t_C} \wedge \\ \quad \exists x : x^s = \bigvee_{\lambda^C(\tau, o)} \tau \in x \wedge x \leq x^t \end{array} \tag{7}$$

The context-dependence of the relation Λ^C_R is reflected by the fact that the spatial as well as the temporal parameters range over regions belonging to levels of granularities associated to the underlying context. The context-dependence of the relation Λ^C_P is reflected by the fact that the temporal parameter ranges over regions belonging to such a level of granularity and the fact that the spatial parameter ranges over mereological *sums* of regions belonging to such a level of granularity.

5 Spatio-temporal regions and relations

5.1 Objects and regions

We represent (four-dimensional) spatio-temporal regions as pairs of temporal and spatial regions, $x = (x^t, x^s)$. In the case of continuants the pair $x = (x^t, x^s)$ specifies the *exact* spatio-temporal region of the continuant o if and only if either $L_R(o, x^t, x^s)$ or $L_P(o, x^t, x^s)$. It specifies the *approximate* spatio-temporal region of the continuant o if and only if either $\Lambda^C_R(o, x^t, x^s)$ or $\Lambda^C_P(o, x^t, x^s)$.

Consider, for example, the movement of the continuant John from London to New York. In this case x^t is the time-period over which John flies and x^s is the path along which John moves from London to New York, i.e., $\Lambda^C_P(John, x^t, x^s)$.

Occurrents (events and processes) do not change location – they just occur. Consequently for occurrents only the notions L_R and Λ^C_R are relevant. An event e is (approximately) located at a spatio-temporal region which is specified by the pair $x = (x^t, x^s)$, i.e., $\Lambda^C_R(e, x^t, x^s)$, if and only if there is a continuant on which e depends such that $\Lambda^C_R(\textit{continuant-of}(e), x^t, x^s)$. Consider the event of John's flight to New York. In this case x^t is the time-period over which the flight occurs and x^s is the path along which the continuant John moves from London to New York.

A process p is (approximately) located at a spatio-temporal region which is specified by the pair $x = (x^t, x^s)$, i.e., $\Lambda^C_R(p, x^t, x^s)$, if and only if there is a continuant on which p depends such that $\Lambda^C_P(\textit{continuant-of}(p), x^t, x^s)$. Consider the process *John's flying to New York*. In this case x^t is the time-period over which the process of flying occurs and x^s is the path along which the continuant John moves from London to New York.

That the event *John's flight* and process *John's flying* are located at the same spatio-temporal region specified by $x = (x^t, x^s)$ is due to the fact that event and process represent two different views of the same change of location of the continuant John. Those different views, however, cannot be taken simultaneously in the same context.

5.2 Relations

Given the representation of spatio-temporal regions as pairs of spatial and temporal regions the next natural thing to do is to define relations between them. The obvious way of doing this is to define relations between two spatio-temporal regions as a *pair* of temporal and spatial relations based on relations between their spatial and temporal components.

Consider two spatio-temporal regions $x_1 = (x^t_1, x^s_1)$ and $x_2 = (x^t_2, x^s_2)$. We define identity and overlap-sensitive relations based on distinguishing identity (=), proper overlap which excludes identity but includes containment (o), and non-overlap (ø) relations among spatial regions and among temporal regions. This gives raise to nine combinatorial

possible spatio-temporal relations: [3]

$R^{st}(x_1, x_2)$	1	2	3	4	5	6	7	8	9
$R^t(x_1^t, x_2^t)$	=	=	=	o	o	o	ø	ø	ø
$R^s(x_1^s, x_2^s)$	=	o	ø	=	o	ø	=	o	ø

(8)

In order to interpret these formal definitions we need to take into account that a given pair $x = (x^t, x^s)$ can specify: (a) the location (the place) at/in which the continuant c is at rest ($L_R(c, x^t, x^s)$ or $\Lambda_R^C(c, x^t, x^s)$); and (b) the region within (the path along) which the continuant c changes location (moves), i.e., $L_P(c, x^t, x^s)$ or $\Lambda_P^C(c, x^t, x^s)$). This will be discussed in the Sections 6, 7, and 8.

6 Relations between occurrents and continuants at rest

6.1 Same time – same place ...

Consider two spatio-temporal regions $x_1 = (x_1^t, x_1^s)$ and $x_2 = (x_2^t, x_2^s)$ within which the continuants o_1 and o_2 are at rest or within which the occurrents o_1 and o_2 occur (or any other combination of occurrents and continuants at rest). We will use the notion of place in order to refer to spatial locations where certain continuants are at rest and where certain events occur. The relation defined in Table 8 are now interpreted as follows: (1) - same-time-same-place (stsp); (3) - same-time-different-place (stdp); (7) - different-time-same-place (dtsp); (9) - different-time-different-place (dtdp).

Consider the following examples: (a) Imagine that the continuants John and Mary are on the same airplane from London to New York. The events John's flight and Mary's flight to New York occur in the same place and at the same time. (b) The processes of John's and Mary's flying to New York occur in the same place and at the same time. (c) The philosophers Marx and Engels did much of their early work while being together in London (same-time-same-place). Later they were able to do collaborate while being in different places at the same time, with Engels in Manchester and Marx in London (same-time-different-place). (d) In order to catch the flu from another person both must be in the same place at the same time. (e) John and Mary missed their date because they were in different places at the agreed time, i.e., John was stuck in a traffic jam while Mary was waiting in the agreed place. (f) John called Mary and they agreed to meet in a different place at a different time (e.g., next day at the McDonald's by the interstate ramp). (g) When they finally met (same-place-same-time) they learned that they could not stand each other and avoided each other from this moment, thereafter they were always at different places at the same time.

[3] There have been defined more detailed sets of spatio-temporal relations, e.g., (Bennett, Cohn, Wolter & Zakharyaschev To appear) or (Hazarika & Cohn 2001). The set of relations considered here, however, proves to be particularly useful for the discussion of the reference to spatio-temporal relation in judgments.

6.2 Judgment, context, and reference to relations

Obviously, one can find plenty of examples of sentences that make use of the spatio-temporal relations corresponding to the cases 1,3,7,9 in Table 8. However these relations are only a subset of the combinatorially possible cases. There do not seem to be equally good examples of sentences that make use of spatio-temporal relations that are defined by the cases 2, 4, 5, 6, 8 in Table 8. Those cases involve spatial or temporal overlap $x_1^t \; o \; x_2^t$ or $x_1^s \; o \; x_2^s$.

From Equation 6 it follows that the regions which satisfy the relation Λ_R^C in a context, C, are mutually disjoint since they belong to a single level of granularity. Consequently, when considering relations between those regions (x_1^t, x_1^s) and (x_2^t, x_2^s) with $x_1^t, x_2^t \in \mathcal{L}_{\mathcal{G}_C^t}$ and $x_1^s, x_2^s \in \mathcal{L}_{\mathcal{G}_C^s}$ only the patterns 1, 3, 7, and 9 in Table 8 can occur.

Of course, there do exist events and processes that do overlap (without being identical). Furthermore all relations given in Table 8 do actually occur between certain occurrents in reality and, hence, can potentially be referred to in sentences regarding these matters. However it is important to distinguish between the occurrence of relationships in reality and the reference to those relations in language. Moreover it is critical to consider *judgments* rather than sentences (Smith & Brogaard 2001).

A judgment is a sentence uttered by a judging subject (the judger) in a certain context. [4] Consequently, when considering reference to spatio-temporal relations it is critical to take the context-dependence of approximate location into account. Approximate location in a given context is determined by the spatial and temporal levels of granularity and the approximate localization function associated to the context. At the formal level this is reflected by Equation 6 leading to the fact that reference to relations determined by the patterns 2, 4, 5, 6, and 8 in Table 8 does not occur in judgments that are made in naturally occurring contexts.

6.3 Choosing levels of granularity

We now discuss how judging subjects choose levels of granularity. We concentrate hereby on the spatial case. In this context we make the following assumptions: (i) Judging subjects have a certain degree of freedom in choosing systems and levels of granularity; (ii) Judging subjects as-

[4] We limit ourselfs to judgments that can be uttered in naturally occurring contexts and admit the possibility to construct 'strange' counter examples in certain circumstances.

sociate the chosen level of granularity to the context before uttering the judgment in question; and (iii) There is no vagueness involved and the judging subjects have complete knowledge in the sense that there is no need to choose a coarse level of granularity due to vagueness of lack of knowledge.

Imagine that John and Mary's second date was arranged to take place in a certain downtown bar. Consider the time while Mary waited on her table for John to arrive and consider the following scenarios : (a) John was the whole time while Mary was waiting on her table stuck in the restroom due to his diarrhea; and (b) John was sitting on Mary's table.

Assume a system of granularities containing a cell 'the downtown bar' (B) with subcells guest room (G) and restroom (R) where the guest room has a separate subcell for the neighborhood of each table (T_i). In judgments then reference to the following spatio-temporal relations can be made:

J_1	*John stsp Mary*	$\Lambda_R(John, B)$	$\Lambda_R(Mary, B))$
J_2	*John stdp Mary*	$\Lambda_R(John, G)$	$\Lambda_R(Mary, R))$
J_3	*John stsp Mary*	$\Lambda_R(John, T_i)$	$\Lambda_R(Mary, T_i))$

Assume now that the judgments, J_i, are made in a context where the judger has the intention to specify whether or not John and Mary have met. In situation (a) the judgments J_1 and J_2 can potentially uttered truly and in situation (b) the judgments J_1 and J_3 can potentially uttered truly. However in situation (a), given the intention of the judger, she will choose the level of granularity allowing her to distinguish the guest room from the restroom and she will judge J_2 rather than J_1. In situation (b) on the other hand she will judge J_3 rather than J_1.

It is important to emphasis the judger has the freedom and the obligation to choose the level of granularity in such a way that the judgment is true either in situation (a) or in situation (b) but not in both. This is because given the freedom to choose an appropriate level of granularity – why would a judger who's intension is to make a meaningful judgment (rather than to make a joke or to utter nonsense) use choose a level of granularity such that a judgment (implied by J_i) about whether or not John and Mary have met is subject to truth-value indeterminacy? Since there are no such reasons in naturally occurring contexts we hold that levels of granularity are chosen in such a way that the resulting judgments are meaningful and determinate.

7 Movement along paths

Consider the relation $\Lambda^C_P(o, x^t, x^s)$ and assume that it holds for the object o, the temporal region x^t and the spatial region x^s. As discussed in Equation 7 the region x^t is an element of a level of granularity $\mathcal{L}_{\mathcal{G}^t_C}$ belonging to a system of granularities $\mathcal{G}^t_C$ and the region x^s is the mereological sum of elements of a level of granularity $\mathcal{L}_{\mathcal{G}^s_C}$ belonging to a system of granularities $\mathcal{G}^s_C$.

In our examples so far we (intuitively) always interpreted the region x^s as a path of movement rather than as the sum of regions occupied over a sequence of arbitrary changes of location. Let us now more carefully distinguish movement from arbitrary change of location like change of shape, growth and shrinking. Let $x_1, \ldots, x_n$ be (pair-wise non-identical) regions at which the continuant o was exactly located at $\tau_1 \ldots \tau_n$. We say that the change of location is movement if and only if the joint intersection of all those regions is empty. [5]

In Definition 7 we demanded that the regions summing up x^s need to belong to the same level of granularity. Since regions forming a single level of granularity are pair-wise disjoint it follows that for change of location, which is not movement in the sense defined above, the notions Λ^C_P and Λ^C_R coincide in the sense that given a fixed level of granularity we have $\Lambda^C_P(o, x^t, x^s)$ if and only if $\Lambda^C_R(o, x^t, x^s)$. Consequently it is sufficient to consider the approximation of location of continuants that move along paths.

7.1 Same path – same time ...

Let the spatio-temporal regions $x_1 = (x^t_1, x^s_1)$ and $x_2 = (x^t_2, x^s_2)$ be the motion-*paths* along which the continuants o_1 and o_2 change location (move). In the case of approximate location the relation $\Lambda^C_P(o, x^t, x^s)$ holds and x^t belongs to a level of granularity and x^s is the sum of regions belonging to some level of granularity.

The relations 1, 3, 7, 9 in Table 8 are now interpreted as follows: (1) - same-time-same-path; (3) - same-time-different-path; (7) - different-time-same-path; (9) - different-time-different-path.

Consider the following examples: (a) The flying of the continuants John and Mary from Chicago to New York in the same airplane is an example for same-time-same-path. (b) If John flew on Monday and Mary flew on Tuesday then this would be an example for different-time-same-path. (c) If John flew to New York on Monday morning and Mary flew to Los Angeles the same morning then between their flying the relation same-time-different-path would hold. (d) If John flew to New York on Monday and Mary flew to Los Angeles on Tuesday, then this would be an example for different-time-different-path.

[5] This is a very rough definition and it is certainly possible to construct counter-intuitive examples (e.g., a car that moves only a few inches). However the definition does capture the intuition that if something moves then usually its end-location is disjoint from the one it started from.

Of course, for each relation in Table 8 there are continuants in reality which change in such a way that those relations hold. Consider the case (4). An example would be two continuants moving (e.g., two cars) along the same path and the time of their journey overlaps without being identical (e.g., one started earlier or one finished earlier or ...). Consider the cases (2) and (5). Examples are such that the paths of two moving occurrents do *cross* each other. For example consider the paths of airline-passengers with connecting flights who share the same airplane for a part of their journeys.

But, again, it is important to distinguish between the occurrence of relationships in reality and the reference to relations in judgments. For example, it is hard to imagine a context in which a judging subject (the judger) would use (4) in order to actually make a judgment. For example, in the case of a car-race she would rather say that one car finished earlier than the other (and thus won). In this case she uses a set of relations with a *finer granularity* at the temporal level than the relations in Table 8. She might use, for example, the Allen relations (Allen 1983), which allow her to distinguish: relations like starts, finishes, or during which are refinements of proper overlap as defined above.

The same argument holds in the case the relations (2) and (5). A judging subject would use relations of a finer level of granularity in order to make judgments. She would use spatio-temporal relations in order to distinguish continuants moving along forking paths, merging paths, or properly crossing paths. (These relations, of course, are refinements of the cases (2) and (5).)

7.2 Refined relations

Consider Figure 2 which shows a number of paths $p_1, \ldots p_4$ in a street network with block structure. Along those paths move the continuants $o_1 \ldots o_4$ over the time-period x^t. The street network forms a system of granularities with cells: 1st, 2nd, 3rd, 4th street; a, b, c avenue; block segments $(1, r_1)$, $(1, r_2)$, ..., (a, c_1), (a, c_1), etc. Those cells form levels of granularity in the obvious way. We call the level of granularity which is formed exclusively by segments *the block level* $\mathcal{L}_B$.

In the context $\mathcal{C}$ we specify approximate spatial location at block level, e.g., $\Lambda^{\mathcal{C}}_P(o_1, x^t, p_1)$, ..., $\Lambda^{\mathcal{C}}_P(o_4, x^t, p_4)$ with $p_1 = (a, c_2) + (2, r_2) + (b, c_3) + (3, r_3)$ and $p_2 = (b, c_2) + (b, c_3) + (b, c_4)$.

In order to understand why in the case of judgments about spatio-temporal relations between continuants moving along paths a finer set of relations is used than the one given in Table 8 we need to take three properties of paths into account: (i) Paths like $p_1 = g_1 + \ldots + g_n$ and $p_2 = h_1 + \ldots + h_m$ (with $g_i, h_j \in \mathcal{L}_B$) are formed by sets of cells which elements can potentially overlap; (ii) Paths like p_1 and p_2 have a mereo-*topological* structure in the sense that some cells are externally connected; (iii) Paths like p_1 and p_2 have an internal *ordering* structure in that sense that connected cells form chains with beginnings and endings.

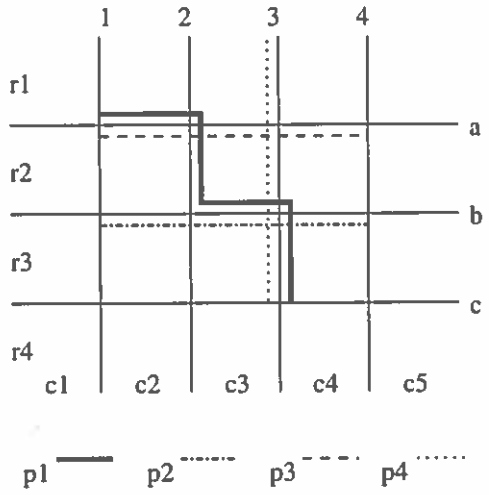


Figure 2: The paths $p_1 \ldots p_4$ in a city block network formed by 1st, 2nd, 3rd, 4th street and a, b, c avenue.

Mereological overlap. Consider the spatio-temporal regions (x^t, p_1) and (x^t, p_2). The spatial regions p_1 and p_2 are mereological sums of cells of a level of granularity rather than single cells in that level, e.g., $p_1 = (a, c_2) + (2, r_2) + (b, c_3) + (3, r_3)$ and $p_2 = (b, c_2) + (b, c_3) + (b, c_4)$. Those sums of cells can potentially overlap in the sense that, when considered as sets (e.g., $\{(a, c_2), (2, r_2), (b, c_3), (3, r_3)\}$), they share constituting cells. For example, in Figure 2 we have $(p_1 \; o \; p_2)$, $(p_1 \; o \; p_3)$, and $(p_1 \; o \; p_4)$. [6]

However, as argued above, in judgments about relations between paths of movement humans usually would make finer distinctions by saying that the paths p_1 and p_2 cross each other, while the paths p_1 and p_3 fork and the paths p_1 and p_4 merge. These distinctions are possible at the level of approximations because in most contexts, given paths of the form $p_1 = (a, c_2) + (2, r_2) + (b, c_3) + (3, r_3)$, we can define an ordering $(a, c_2) \prec (2, r_2) \prec (b, c_3) \prec (3, r_3)$. In order to see this we need to take topological and ordering structure into account.

Topological structure. Levels of granularities that are used to specify approximate location of moving continu-

[6] Notice that the overlap of the paths p_2 and p_4 as well as the overlap of the paths p_3 and p_4 cannot be referred to at the block level of granularity since there are no shared cells.

ants are *full* in the sense that their mereological sum is identical to the next superordinate unit in the underlying system of granularity (Section 4.2). If we assume that cells forming systems of granularity are topologically simple then it follows that cells forming a level of granularity are either externally connected or they are disconnected in the sense that they either do share boundary parts or they do not. If we assume continuous movement then it follows that paths of movement are formed by chains of externally connected cells.

Ordering structure. We now assume that paths of movement correspond to those chains which are not self-intersecting, i.e., we omit cases where the continuant under consideration drives around the block in search for a parking spot. The order in which the cells appear in chains such as $(a, c_2) \prec (2, r_2) \prec (b, c_3) \prec (3, r_3)$ corresponds to the order in which the continuant in question was located at them at subsequent periods of time. We now can distinguish the begin and the end of a chain of cells as the minimal and the maximal element of a chain with respect to the ordering $\prec$.

Let p_i and p_j be two paths of movement such that $\Lambda^C_P(o_i, x^t, p_i)$ and $\Lambda^C_P(o_j, x^t, p_j)$. We say that (a) p_i and p_j fork if and only if $p_i \neq p_j$ and $\min(p_i) = \min(p_j)$; (b) they merge if and only if $p_i \neq p_j$ and $\max(p_i) = \max(p_j)$; and (c) they cross if and only if $p_i \cap p_j \neq \emptyset$ and $\{\min(p_i), \min(p_j), \max(p_i), \max(p_j)\} \cap (p_i \cap p_j) = \emptyset$.

Obviously, those definitions only account for simple situations, however we suggest that due to the feature of granularity it is those simple situations that are referred to in judgments. Also, if $p_i \subset p_j$ then we can distinguish the Allen-relations relations p_i *starts* p_j, p_i *finishes* p_j, and p_i *during* p_j in a similar fashion.

8 Relations between places and paths

Consider two spatio-temporal regions $x_1 = (x_1^t, x_1^s)$ and $x_2 = (x_2^t, x_2^s)$ representing spatio-temporal location. Let x_1 be the location (place) of rest of a continuant or the place at which an event or process occurs, i.e., $\Lambda^C_R(o_1, x_1^t, x_1^s)$, and let x_2 be the motion-path of the continuant o_2, i.e., $\Lambda^C_P(o_2, x_2^t, x_2^s)$. The relations in Table 8 are now interpreted as antisymmetric relations with places as first and paths as second argument. The patterns are interpreted as follows: (2,5) - place-on-path(-at-the-same-time), (8) place-on-path-at-different-time, (3,6,9) - place-not-on-path. Hereby (2,5) abbreviates that either pattern 2 or 5 holds.

Consider the following examples. *place-on-path*: The oil-slick on the path of the nuclear waste transport; the speed trap on the path of your journey home. *place-on-path-at-different-time*: the oil-slick that was removed from the road before the nuclear waste transport took this path. *place-not-on-path*: The traffic jam on route I95 while you are traveling home on route I90; or The North Pole and your journey to the South Pole.

In order to justify this grouping and this interpretation of the patterns in Table 8 we need to discuss the spatial and the temporal dimension separately.

8.1 The spatial dimension

Consider the relations in Table 8. The use of spatio-temporal relations in judgments about spatio-temporal relationships between places and paths[7] seems to be limited to the mentioned prototypical cases. The cases 1, 4, and 7 defined in Table 8 do not occur in judgments about relationships between places and path. This is because the spatial extension of the places and the paths, that are involved in those relations, must be (significantly) distinct in size. To see this consider the following:

The special case of spatial co-location of place and path occurs in cases similar to the following. Consider the spatial location of the event 'John's flight to New York', the process 'John's flying to New York', and the movement of the continuant 'John' from London to New York. Here pattern 1 in Table 8 does potentially hold. However the event 'John's flight', the process 'John's flying', and the movement of the continuant'John' represent distinct views onto the same continuant (John) within the same time-period. However only *one* of those views can be taken in a single context. Consequently patterns 1, 4, and 7 in Table 8 cannot be used to specify a relation in a judgment which is bound to a single context.

In general the spatial extension of the place is (much) smaller than the extension of the path, i.e., the maximal diameter of the place is (much) smaller than the length of the path. Given the definitions of Λ^C_R and Λ^C_P (Equations 6 and 7) this can be understood as follows. A judgment is made in a specific context and within that context we have a *single* level of granularity for spatial reference. Cells forming places as well as the cells summing up paths belong to this level. Consequently, paths formed by a *multitude* of cells therefore must be (significantly) larger than the place formed by a single cell. (We assume that cells at same level of granularity are of roughly the same size.)

Notice that natural language seems to make additional distinctions regarding the scale of the difference of the size

[7] We use the phrase 'relation between place and path' as an abbreviation for 'relation between the place x_1^s at which the event o_1 occurs (the continuant o_1 is at rest) over the interval x_1^t and the path of movement, x_2^s, of a continuant o_2 over the period of time x_2^t'.

between places and path by distinguishing relationships like place-*on*-path ('There was an oil-slick on his path') or place-*along*-path ('There was a fire along his path') in the sense that the underlying level of granularity in the case of place-*on*-path is much finer than in the case of place-*along*-path.

There are cases where the spatial extension of the place is (much) larger than the extension of the path. In those cases the path is for example *in* the place as in 'When inside the Federal Building take the elevator to the 13th floor'. In this case, however, the place is used in order to specify the approximate location of the path in the sense of $\Lambda^{\mathcal{C}}_{R}$ and we are dealing with a different set of relations.

8.2 The temporal dimension

Consider the definitions of place-on-path(-at-the-same-time) and place-not-on-path. At the temporal level we allow for $x^t_1 = x^t_2 \vee x^t_1\ o\ x^t_2$ in the case of place-on-path and for $x^t_1 = x^t_2 \vee x^t_1\ o\ x^t_2 \vee x^t_1\ \emptyset\ x^t_2$ in the case of place-not-on-path. This indicates that the relations in Table 8 are *too fine* in granularity in order to be used directly in judgments about relations between paths and places.

Consider the judgment 'the oil-slick is on the path of the nuclear waste transport'. For the binary relation 'oil-slick' place-on-path 'path of the nuclear waste transport' to hold the existence of the oil-slick must overlap temporally the existence of the spatio-temporal object 'path-of the nuclear waste transport'. Overlap here is meant in a sense that includes identity. The oil-slick might have existed before and might continue to exist afterwards. It might be placed on the road after the transport has begun its journey. It might even have been removed by a security team before the transport has actually arrived at this particular place. Similar examples can be found for the relation place-not-on-path.

9 Discussion

In an attempt to understand the way people make judgments about relations between spatio-temporal objects it is important to understand the role of systems and levels of granularity utilized in the process of judging.

Judgments about spatio-temporal relations are based on the specification of approximate spatio-temporal location which is based on reference to hierarchically organized systems of cells forming levels of granularity. When making a certain judgment the judging subject has a certain degree of freedom to chose a system and a particular level of granularity which she considers appropriate in order a judgment in a *determinate* fashion and in order to convey useful information. However the *degree* of freedom of choice is different in the case of (a) specifying location events and processes or continuants at rest using the relation $\Lambda^{\mathcal{C}}_{R}$ and in the case (b) of specification of change of location of a continuant along a path by means of the relation $\Lambda^{\mathcal{C}}_{P}$.

If we assume that judgments are made in such a way that they convey *determinate and useful information* then we can state the thesis that the more freedom the judging subject has in choosing an appropriate level of granularity the less relations are needed in order to specify spatio-temporal relations in a determinate manner. On the other hand the more restrictive the choice of level of granularity the richer the vocabulary that is needed in order to refer to spatio-temporal relations in a determinate fashion.

In other words there are two different strategies that judging subjects employ in order to make judgments that convey determinate and useful information: The adjustment of system and level of granularity of approximation of location in case (a) versus the refinement of the granularity of the relations in case (b). The remaining question now is why judging subjects have different degree of freedom of choosing system and level of granularity of approximation.

The answer to this question can be found by analyzing the properties of the systems and levels of granularities underlying the judgments in question. Consider systems of granularities formed by hierarchically organized places that underlie judgments of type (a). Those systems of granularity have only a weak structure in the sense that they usually lack the property of fullness (Section 4.2) and, hence, also lack topological and ordering structure in the sense discussed in Section 7.2.

Due to the lack of structural properties the judging subject has the freedom to choose subsystems (like the one in Figure 1) and levels of granularity within them in ways which she considers to be appropriate without having to worry about preserving structural properties such as fullness, topology, and ordering. The advantage of those systems of granularities is that (1) they are easy to construct, and (2) they contain only cells which are relevant in the context at hand. This flexibility allows the judging subject to choose a level of granularity in such a way that only a few relations (a small vocabulary) such as same-time-same-place etc. (the relations corresponding to the pattern 1,3,7,9 in Table 8) are needed in order to make judgments in the required determinate fashion as discussed in Section 6.3.

Systems of granularities that underly judgments of type (b) are usually full and have sophisticated topological and ordering structure as discussed in Section 7.2. Those systems of granularity are in their structure firmly grounded in reality and therefore are less flexible in the sense that they are either used in judgments as they are or not used at all. On the other hand, their sophisticated structure allows for a

rich vocabulary for specifying spatio-temporal relations as demonstrated in Section 7.2.

10 Related work

Qualitative spatio-temporal relations have been discussed widely in the literature, e.g., (Claramunt, Theriault & Parent 1997, Hazarika & Cohn 2001, Bennett et al. To appear). (Hazarika & Cohn 2001) propose a spatio-temporal logic between four-dimensional regions based on the notions of spatial, temporal, and spatio-temporal connectedness. (Bennett et al. To appear) proposed a multi-dimensional modal logic for spatio-temporal reasoning as a temporalized version of the region connection calculus (RCC). Consequently, at the spatial level they distinguish the eight RCC relations and at the temporal level they distinguish the thirteen Allen relations (Allen 1983, Randall et al. 1992). A similar set of relations was defined in (Claramunt et al. 1997) in a point-set theoretical framework employing Egenhofer's intersection model (Egenhofer & Franzosa 1991). Within those approaches spatio-temporal relations between *occurrents* based on their *exact* location can be represented.

(Stell 2001) takes the approximate nature of spatio-temporal location into account and defines qualitative spatio-temporal extents like somewhere-sometime, always-somewhere, etc. which are based on specifications of location with respect to full levels of granularity.

The issue of granularity was discussed in spatial and temporal contexts for example in (Bettini et al. 1998, Euzenat 1995, Stell & Worboys 1998), however mostly in full systems of granularity and not in the context of reference to relations in judgments. Spatial and temporal relations in the context of approximate spatial and temporal location were discussed in (Bittner & Stell 2000), (Bittner to appear). The present paper extends this work to the spatio-temporal domain, to non-full systems of granularities, and by explicitly taking the notion of context into account.

11 Conclusions

We discussed qualitative relations between spatio-temporal objects based on their location in space and time. The location of a spatio-temporal object in a region of space or time may be exact or approximate. Exact location emphasizes the exact fit between object and region. A spatio-temporal object is located approximately in a region of space or time if its exact region is a part of this region. There are multiple ways of approximating exact location at different levels of granularity.

Critical for the understanding of qualitative relations between spatio-temporal objects is the distinction between continuants and occurrents and the closely related distinction between rest and change. Only continuants are capable of change and hence of rest. Occurrents do not change. They just occur and characterize modes of rest or change of occurrents. The spatio-temporal occurrence of events and processes is closely related to the notion of place in the sense that events and processes occur in places. The change of location of continuants takes place along paths.

We defined and discussed relations between various kinds of spatio-temporal objects. This discussion showed that the formalization of spatio-temporal relations needs to take into account the notion of context. While people seem to deal effortlessly with the different meanings of relations like same-place-same-time in varying contexts, it is hard to formalize the context dependence that is thereby involved. We addressed the problem of context by distinguishing relationships that hold between spatio-temporal objects from the reference to spatio-temporal relations in judgments made in specific contexts.

A judgment in a specific context is characterized by: (a) a statement about spatio-temporal relation between two objects; (b) the types of those objects (continuants or occurrents); (c) a certain level of granularity at which the approximate location of the two objects is specified; (d) a certain level of granularity of the particular spatio-temporal relations that are used in this judgment. The levels of granularity of approximate location and spatio-temporal relations are chosen by the judging subject in such a way that the resulting judgment provides determinate and useful information (is not subject to truth-value indeterminacy).

The relationship between the level of granularity of approximate location and the level of granularity of the spatio-temporal relations at hand is quite complex. We showed that: (1) In contexts where a judgment was made about relationships between (among) occurrents or continuants at rest, the level of granularity of approximate location is chosen in such a way that a small number of relations at an intermediate level of granularity is used (same-place-same-time, and so on). (2) In contexts where a judgment is made about relationships between changing continuants, the level of granularity of approximate location is relatively fixed and cannot be chosen by the judging subject as freely as in the previous case. However in those contexts a finer set of spatio-temporal relationships is used in order to make the distinctions that are necessary in order to make a determinate judgment.

Ongoing research is directed towards giving a more formal account of the context-dependence involved in judgments about spatio-temporal relations. This work is based on the theory of granular partitions proposed in (Bittner & Smith 2001*a*) and (Bittner & Smith 2001*b*).

Acknowledgements

Thanks are due to Barry Smith, Stephan Winter, and one of the anonymous reviewers for their helpful comments. The support by the Alexander von Humboldt Stiftung under the auspices of its Wolfgang Paul Programme, DARPA under the Rapid Knowledge Formation program, and the National Science Foundation under the Research on Learning and Education program is gratefully acknowledged.

References

Allen, J. (1983), 'Maintaining knowledge about temporal intervals', *Communications of the ACM* **26**(11), 832–843.

Bennett, B., Cohn, A., Wolter, F. & Zakharyaschev, M. (To appear), 'Multi-dimensional modal logic as a framework for spatio-temporal reasoning', *Applied Intelligence*.

Bettini, C., Wang, X. & Jajodia, S. (1998), 'A general framework for time granularity and its application to temporal reasoning', *Annals of Mathematics and Artificial Intelligence* **22**, 29–58.

Bittner, T. (to appear), 'Approximate temporal reasoning', *Annals of Mathematics and Artificial Intelligence*.

Bittner, T. & Smith, B. (2001*a*), A taxonomy of granular partitions, *in* D.R.Montello, ed., 'Spatial Information Theory, COSIT '01', Vol. 2205 of *Lecture Notes in Computer Science*, Berlin/New York: Springer, pp. 28–43.

Bittner, T. & Smith, B. (2001*b*), 'A theory of granular partitions', *Department of Computer Science, Northwestern University*.

Bittner, T. & Stell, J. (2000), Rough sets in approximate spatial reasoning, *in* 'Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing (RSCTC'2000)', Lecture Notes in Computer Science (LNCS), Springer-Verlag.

Casati, R. & Varzi, A. (1995), 'The structure of spatial localization', *Philosophical Studies* **82**(2), 205–239.

Claramunt, C., Theriault, M. & Parent, C. (1997), A qualitative representation of evolving spatial entities in two-dimensional spaces, *in* S. Carver, ed., 'Innovations in GIS V', Taylor & Francis, pp. 119–129.

Egenhofer, M. J. & Franzosa, R. D. (1991), 'Point-set topological spatial relations', *International Journal of Geographical Information Systems* **5**(2), 161–174.

Euzenat, J. (1995), An algebraic approach for granularity in qualitative space and time representation, *in* 'Proceedings of the IJCAI-95', pp. 894–900.

Hazarika, S. M. & Cohn, A. G. (2001), Qualitative spatio-temporal continuity, *in* D. R. Montello, ed., 'Spatial Information Theory: Foundations of Geographic Information Science; Proceedings of COSIT'01', Vol. 2205 of *LNCS*, Springer Verlag, pp. 92–107.

Randall, D., Cui, Z. & Cohn, A. (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.

Rigaux, P. & Scholl, M. (1995), Multi-scale partitions: Application to spatial and statistical databases, *in* M. Egenhofer & J. Herrings, eds, 'Advances in Spatial Databases (SSD'95)', number 951 *in* 'Lecture Notes in Computer Science', Springer-Verlag, Berlin.

Simons, P. (1987), *Parts, A Study in Ontology*, Clarendon Press, Oxford.

Smith, B. & Brogaard, B. (2001), 'A unified theory of truth and reference', *Logique et Analyse*.

Smith, B. & Brogaard, B. (to appear), 'Quantum mereotopology', *Annals of Mathematics and Artificial Intelligence*.

Stell, J. & Worboys, M. (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. (2001), Qualitative extents for spatio-temporal granularity, *in* 'Workshop on Spatial Vagueness, Uncertainty and Granularity'.

Scientific Benchmarking with Temporal Logic Decision Procedures

Ullrich Hustadt
Department of Computer Science
University of Liverpool
Liverpool L69 7ZF, UK
U.Hustadt@csc.liv.ac.uk

Renate A. Schmidt
Department of Computer Science
University of Manchester
Manchester M13 9PL, UK
schmidt@cs.man.ac.uk

Abstract

In this paper we propose a hypothesis-driven design of the empirical analysis of different decision procedures which we refer to as *scientific benchmarking*. The approach is to start by choosing the benchmark problems for which, on the basis of analytical considerations, we expect a particular decision procedure to exhibit a behaviour different from another decision procedure. Then empirical tests are performed in order to verify the particular hypothesis concerning the decision procedures under consideration. As a case study, we apply this methodology to compare different decision procedures for propositional temporal logic. We define two classes of randomly generated temporal logic formulae which we use to investigate the behaviour of two tableaux-based temporal logic approaches using the Logics Workbench, a third tableaux-based approach using the *STeP* system, and temporal resolution using a new prover called *TRP*.

1 Introduction

In recent years it has become common to compare decision procedures for a wide variety of logics, including Boolean logic, quantified Boolean logic, basic modal logic, and description logics on sets of randomly generated classes of formulae [1, 3, 7, 11, 16, 17, 18, 24, 25, 27]. Although this form of analysis of algorithms and their implementations has lead to a number of interesting results and has certainly contributed to recent progress in the sophistication of decision procedures for the logics under consideration, empirical analysis with randomly generated problem sets is not without problems.

One of the main problems stems from the fact that underlying any class of randomly generated problems is a probability density function μ which assigns probabilities to instances of the problem class and the fact that the analysis is sensitive to the choice of μ. The most famous example illustrating this problem is the density function underlying a result by Goldberg [12, 13]. Goldberg has shown (analytically) that the satisfiability problem of Boolean formulae can be solved by a DPLL procedure in polynomial time in the average case. Subsequently, Franco and Paull [10] proved that this result is due to the density function he assumed and not a particular feature of the DPLL algorithm, and that for a more reasonable density function on the class of Boolean formulae, a variant of the DPLL procedure needs exponential time in the average case.

A related problem is that it is the aim of most comparisons to demonstrate differences between various algorithms, data structures, or implementation techniques. They are often used to illustrate the usefulness of a new idea by comparing an implementation A, incorporating this idea, with an implementation B that does not. However, even if we consider a problem class together with a probability density function μ that makes the problem class reasonably hard in the average case, there is no guarantee that we will be able to see any differences in the average performance of implementations A and B. It may well be that the particular problems on which A is superior to B have a low probability with respect to μ and can therefore not influence the average performance sufficiently to have a noticeable impact. This might be justified if μ models the probability of problems close to what we expect in the 'real world', but this seems to be rarely the case for the logics mentioned in the beginning and the density functions described in the literature.

The problem is aggravated if we consider implementations A and B of decision procedures which vary in more aspects than one. In this case, even if we can

observe a noticeable difference in the average performance, we have to explain which particular aspects do contribute to this difference and which do not. Again, the choice of the density function μ can influence our ability to provide convincing evidence for any such explanation.

Hooker [15] proposes the following experimental design to overcome some of the problems related to this kind of empirical analysis: We should start by setting up a list of hypotheses about factors that could affect the performance of an algorithm. Some of these factors can be related to features of the problems we like to deal with, e.g. size of the problems, a specific structure of the problems, etc. Other factors can be related to the algorithm itself, e.g. its inference rules, its heuristics, its redundancy checks, etc. When formulating a hypothesis we should use some abstract measure to describe the effect of a factor, for example, the number of nodes in a tableau or the length of a refutation using resolution, instead of simply relying on running time. For each factor we should set up a benchmark suite suitable for verifying our hypothesis about the influence of the factor on the performance of the algorithm. The benchmark problems can be purely artificial and need not be related to any 'real world' problems. So randomly generated problems are well suited for this purpose. Of course, we still have to be careful that the generated problems possess the problem characteristics needed for the test. Since the intention is not to perform a competitive test, it is not necessary to have an efficient implementation of our algorithm at hand. The only necessary prerequisite is that we are able to alter the implementation to test our hypotheses about the algorithm itself, for example, we need to be able to turn off specific inference rules and adjust specific heuristics. Hooker calls this kind of experimental design *scientific testing*. It is an implicit assumption of the scientific testing methodology that we restrict ourselves to variations of one basic algorithm.

Of course, this is a considerable restrictions if we are interested in assessing a broad variety of decision procedures for a particular logic. Simulation results relating, for example, derivations in tableau calculi for basic modal logic with those in the translation approach [4, 19, 20] or derivations using the inverse method with the automata approach to modal satisfiability [2], may still form a basis for an abstract measure for the performance of decision procedures based on different calculi. However, if this abstract measure is on the level of inference steps in the calculus underlying each of the decision procedures, then one may argue that this puts certain approaches at a disadvantage, in particular, if they allow the use of specialised data structures and algorithms which result in a much higher inference rate than for competing approaches. In such a case it may also be important that the implementation of the decision procedures under consideration is sufficiently sophisticated to enable each approach to show its full strength. Consequently, we will in most cases have to rely on existing implementations, since otherwise the implementational effort required will be excessive. So, in the absence of an abstract measure relating the performance of different decision procedures, or if we feel that such an abstract measure would lead to an incorrect assessment, measuring the running time of existing implementations of decision procedures for a particular logic can still be a viable option. What remains of the scientific testing methodology is that the benchmark problems are chosen to verify a particular hypothesis concerning the decision procedures under consideration. We call this kind of experimental design *scientific benchmarking* or *hypothesis-driven benchmarking*.

There is a danger that scientific benchmarking might be abused to highlight only the strengths of a particular decision procedure over others. This is not the intention. On the contrary, we believe the aim should be to identify and study both the strengths and weaknesses of decision procedures, to gain insight, and therefore, increase the scientific value of a study. Thus, an important component of scientific benchmarking is the analytical analysis of a given problem class with respect to the features of, and/or methods used in, the decision procedures which are subject of the analysis.

In the following we apply this scientific benchmarking approach to a comparison of decision procedures for propositional linear time logic PLTL. Our aim is to identify some problem sets with characteristic properties for which we expect a particular decision procedure to exhibit a behaviour different from other decision procedures.

For propositional linear time logic PLTL there are three major approaches: (i) tableaux- and sequent-based calculi, (ii) automata-based approaches, and (iii) temporal resolution calculi. In most publications on tableaux-based calculi for PLTL [14, 29] the decision procedure proceeds in two phases: Given a PLTL formula φ which has to be tested for satisfiability, first, the procedure creates a pre-model $\mathcal{M}$ for the formula by applying tableaux expansion rules to φ, then, in the second phase the procedure checks whether $\mathcal{M}$ satisfies all eventuality formulae. Two decision procedures based on this approach have been developed by Janssen [21] and McGuire et al. [22]. A C++ implementation of Janssen's procedure is incorporated

as the 'satisfiability' function in the PLTL module of the Logics Workbench Version 1.1, while the *STeP* system [23] includes an implementation of McGuire's procedure.

In contrast, Schwendimann [26] presents a one-phase tableau calculus which checks locally, on-the-fly, for the fulfilment of eventuality formulae on a branch-by-branch basis. A C++ implementation of this calculus is incorporated as the 'model' function in the PLTL module of the Logics Workbench Version 1.1.

Finally, the resolution method for PLTL proposed by Fisher [8] (see also [5, 6, 9]) involves the translation of PLTL formulae to a clausal normal form, classical resolution within states and temporal resolution over states between eventuality formulae like $\Diamond \neg p$ and formulae that together imply $\Box p$. We have used an experimental implementation, called *TRP* (written in SICStus Prolog 3.8.6), for our experiments. *TRP* is an extension of a first-order resolution theorem prover by temporal resolution.

Since the resolution method for PLTL is less well-established compared to the tableaux-based (and automata-based) approaches, we will focus on highlighting some strengths and weaknesses of this approach compared to the three tableaux-based decision procedures mentioned above. In particular, we will design two classes of randomly generated PLTL formulae, the first class will demonstrate that *TRP* can exhibit better performance than the other approaches, while the second class will demonstrate a shortcoming of *TRP* that allows one of the other approaches to outperform *TRP*.

2 Basics of PLTL

Let P be a set of propositional variables. The set of formulae of *propositional linear time logic* PLTL (over P) is inductively defined as follows: (i) $\top$ is a formula of PLTL, (ii) every propositional variable of P is a formula of PLTL, (iii) if φ and ψ are formulae of PLTL, then $\neg\varphi$ and $(\varphi \vee \psi)$ are formulae of PLTL, and (iv) if φ and ψ are formulae of PLTL, then $\bigcirc\varphi$ (in the next moment of time φ is true), $\Diamond\varphi$ (sometimes in the future φ is true), $\Box\varphi$ (always in the future φ is true), $\varphi \,\mathcal{U}\, \psi$ (φ is true until ψ is true), and $\varphi \,\mathcal{W}\, \psi$ (φ is true unless ψ is true) are formulae of PLTL. Other Boolean connectives including $\bot$, $\wedge$, $\rightarrow$, and $\leftrightarrow$ are defined using $\top$, $\neg$, and $\vee$. A formula of the form $\Diamond\varphi$ is called a $\Diamond$-*formula*. Analogously, for the remaining temporal operators.

We also use $\top$ and $\bot$ to denote the empty conjunction and disjunction, respectively. A *literal* is either p or $\neg p$ where p is a propositional variable.

PLTL-formulae are interpreted over ordered pairs $\mathcal{I} = \langle \mathcal{S}, \iota \rangle$ where (i) $\mathcal{S}$ is an infinite sequence of *states* $(s_i)_{i\in\mathbb{N}}$ and (ii) ι is an *interpretation function* assigning to each state a subset of P.

For an interpretation ι and a state $s_i \in \mathcal{S}$ we can define a *propositional valuation* I by $I(p) = (p \in \iota(s_i))$ for every $p \in \mathsf{P}$. On the other hand, given an infinite sequence $(I_i)_{i\in\mathbb{N}}$ of propositional valuations, we can define an interpretation function ι by $\iota(s_i) = \{p \mid I_i(p) = \text{true}\}$.

We define a binary relation $\models$ between a PLTL-formula φ and a pair consisting of a PLTL-interpretation $\mathcal{I} = \langle \mathcal{S}, \iota \rangle$ and a state $s_i \in \mathcal{S}$ as follows.

$\mathcal{I}, s_i \models \text{true}$
$\mathcal{I}, s_i \models p$ iff $p \in \iota(s_i)$
$\mathcal{I}, s_i \models \neg\varphi$ iff $\mathcal{I}, s_i \not\models \varphi$
$\mathcal{I}, s_i \models \varphi \wedge \psi$ iff $\mathcal{I}, s_i \models \varphi$ and $\mathcal{I}, s_i \models \psi$
$\mathcal{I}, s_i \models \varphi \vee \psi$ iff $\mathcal{I}, s_i \models \varphi$ or $\mathcal{I}, s_i \models \psi$
$\mathcal{I}, s_i \models \bigcirc\varphi$ iff $\mathcal{I}, s_{i+1} \models \varphi$
$\mathcal{I}, s_i \models \Box\varphi$ iff for all $j \in \mathbb{N}$, $j \geq i$ implies $\mathcal{I}, s_j \models \varphi$
$\mathcal{I}, s_i \models \Diamond\varphi$ iff there exists $j \in \mathbb{N}$ such that $j \geq i$ and $\mathcal{I}, s_j \models \varphi$
$\mathcal{I}, s_i \models \varphi \,\mathcal{U}\, \psi$ iff there exists $j \in \mathbb{N}$ such that $j \geq i$, $\mathcal{I}, s_j \models \psi$, and for all $k \in \mathbb{N}$, $j > k \leq i$ implies $\mathcal{I}, s_k \models \varphi$
$\mathcal{I}, s_i \models \varphi \,\mathcal{W}\, \psi$ iff $\mathcal{I}, s_i \models \varphi \,\mathcal{U}\, \psi$ or $\mathcal{I}, s_i \models \Box\varphi$

If $\mathcal{I}, s_i \models \varphi$ then we say φ is *true*, or *holds*, at s_i in $\mathcal{I}$. An interpretation $\mathcal{I}$ *satisfies* a formula φ iff φ holds at s_0 in $\mathcal{I}$ and it *satisfies* a set N of formulae iff for every formula $\psi \in N$, $\mathcal{I}$ satisfies ψ. In this case, $\mathcal{I}$ is a *model* for φ and N, respectively, and we say φ and N are (PLTL-)satisfiable. The satisfiability problem of PLTL is known to be PSPACE-complete [28].

Arbitrary PLTL-formulae can be transformed into a *separated normal form* (SNF) in a satisfiability equivalence preserving way using a renaming technique replacing non-atomic subformulae by new propositions and removing all occurrences of the $\mathcal{U}$ and $\mathcal{W}$ operator [8, 9]. The result is a set of *SNF clauses* of the following form (which differs slightly from [8, 9]).

(initial clause) $\bigvee_{i=1}^{n} L_i$
(global clause) $\Box(\bigvee_{j=1}^{m} K_j \vee \bigvee_{i=1}^{n} \bigcirc L_i)$
(sometime clause) $\Box(\bigvee_{j=1}^{m} K_j \vee \Diamond L)$

Here, K_j, L_i, and L (with $1 \leq j \leq m$, $1 \leq m$, and $1 \leq i \leq n$, $0 \leq n$) denote literals.

Given a PLTL-formula φ the transformation to SNF starts with a set $\{q, \neg q \vee \varphi\}$, where q is a new propositional variable, and exhaustively applies a set of transformation rules to this set. We do not give a complete description, but only present the transformation rules dealing with formulae of the form $\Box\varphi$ and $\varphi \mathcal{W} \psi$ which are important for the understanding of the temporal resolution rule presented below.

$$\{\neg p_1 \vee \Box\varphi\} \Rightarrow \left\{\begin{array}{l}\neg p_1 \vee p_2 \\ \Box(\neg p_2 \vee \bigcirc p_2) \\ \Box(\neg p_2 \vee \varphi)\end{array}\right\}$$

if φ is a literal and p_2 is new.

$$\{\neg p_1 \vee \varphi \mathcal{W} \psi\} \Rightarrow \left\{\begin{array}{l}\neg p_1 \vee \varphi \vee \psi \\ \neg p_1 \vee p_2 \vee \psi \\ \Box\neg p_2 \vee \bigcirc(\varphi \vee \psi) \\ \Box\neg p_2 \vee \bigcirc(p_2 \vee \psi)\end{array}\right\}$$

if φ and ψ are literals and p_2 is new. The result of the transformation of a formula φ to separated normal form is denoted by $\mathrm{snf}(\varphi)$.

We need some additional terminology for SNF clauses. If $C = \Box(\bigvee_{j=1}^{m} K_j \vee \bigvee_{i=1}^{n} \bigcirc L_i)$ is a global clause, then by $\mathrm{now}(C)$ we denote $\bigvee_{j=1}^{m} K_j$ and by $\mathrm{next}(C)$ we denote $\bigvee_{i=1}^{n} L_i$. If $C = \Box(\bigvee_{i=1}^{n} L_i)$, then $\mathrm{now}(C)$ will denote the empty clause and $\mathrm{next}(C)$ will denote $\bigvee_{i=1}^{n} L_i$.

Lemma 1 ([8, 9]). *Let φ be a formula of* PLTL. *Then, φ is* PLTL*-satisfiable if and only if* $\mathrm{snf}(\varphi)$ *is* PLTL*-satisfiable, and* $\mathrm{snf}(\varphi)$ *is computable in time polynomial in the size of φ.*

3 Random PLTL-formulae

The classes we introduce are intended to test the following hypothesis: We conjecture that the temporal resolution calculus will perform better than the three tableau calculi under consideration on problem classes consisting of sets of SNF clauses which impose uniform non-trivial constraints on the states of potential models. On the other hand, we expect the tableau calculi to have an advantage over temporal resolution if a considerable number of clauses impose only trivial constraints on most of the states, particularly, if these clauses include sometime clauses.

The intuition underlying the notion of a clause imposing a trivial or non-trivial constraint on the states is best explained with examples. Consider the PLTL-formula

$$\varphi_1 = \neg p \wedge \Box(\neg p \vee \bigcirc(p \wedge \psi))$$

where ψ is an arbitrary, large PLTL-formula. Suppose we want to construct a model $\mathcal{I} = \langle \mathcal{S}, \iota \rangle$ for φ_1. The first conjunct of φ_1 imposes the constraint that p has to be false in the initial state s_0, that is, $p \notin \iota(s_0)$. But, then the second conjunct of φ_1 imposes no further constraints on s_0. Since $\neg p$ is true at s_0, so is $\neg p \vee \bigcirc(p \wedge \psi)$. The construction of the model can then be completed by defining $\iota(s_{i+1}) = \iota(s_0)$ for all $i \in \mathbb{N}$. In this construction the truth of ψ in any of the states of $\mathcal{I}$ was never considered. Thus, although ψ is a large PLTL-formula by assumption and although it may seem at first sight that the second conjunct of φ_1 imposes a complex constraint on potential models of φ_1, it turns out that it imposes only a 'trivial' constraint since we were able to satisfy the constraint by simply making p false at every state.

Now, consider the PLTL-formula

$$\varphi_2 = \Box(p \vee q) \wedge \Box(\neg p \vee \bigcirc\psi_1) \wedge \Box(\neg q \vee \bigcirc\psi_2)$$

where ψ_1 and ψ_2 are arbitrary, large PLTL-formulae. Again, we want to construct a model $\mathcal{I} = \langle \mathcal{S}, \iota \rangle$ for this formula. In constrast to φ_1, where the first conjunct only imposed a constraint on the first state of the model, here the first conjunct $\Box(p \vee q)$ imposes a uniform constraint on all states of $\mathcal{I}$. If we start the construction at state s_0, then we have to make either p or q true at this state. We can also see that, depending on our choice, we will have to consider the truth of $\bigcirc\psi_1$ or $\bigcirc\psi_2$ which, given that ψ_1 and ψ_2 are large formulae, can turn out to be non-trivial. Furthermore, since $\bigcirc\psi_1$ and $\bigcirc\psi_2$ impose constraints on the next state s_1, which we have not considered yet, we have no indication whether making p true at s_0 will lead to an 'easier' model construction than making q true at s_0 or vice versa. In addition, $\Box(p \vee q)$ enforces that $p \vee q$ has to be true at s_1 as well; a constraint that may interact with ψ_1 or ψ_2 being true at s_1. Thus, in the context of the second and third conjunct of φ, the first conjunct $\Box(p \vee q)$ imposes a non-trivial constraint on our model construction.

Since tableau derivations are closely related to model construction approaches as the one we have used for the two example formulae, it is reasonable to expect that tableau calculi find formulae similar to φ_1 easier than formulae similar to φ_2. Before we explain why we expect that temporal resolution will behave in the opposite way, we first define the classes of formulae with which we are going to test the hypothesis.

Based on Lemma 1 it is clear that in the construction of classes of formulae suitable to test the hypothesis we can restrict ourselves to sets of SNF clauses instead of arbitrary PLTL-formulae. The similarity of

SNF clauses to propositional clauses makes it easy to generalise the standard random generator for uniform propositional kSAT clauses to SNF clauses. This can be done in a number of ways. The random generator we have used in our experiments takes as input the number of propositional variables n, the number of clauses l, a probability p, and a number k.

For the first class of PLTL-formulae, denoted by $\mathcal{C}^1_{ran}$, which will test the first part of the hypothesis, each randomly generated formula φ is a conjunction of SNF clauses of the following form

$$\begin{aligned}
&\Box(\bigcirc L^1_1 \vee \ldots \vee \bigcirc L^1_k) \wedge \ldots \wedge \Box(\bigcirc L^l_1 \vee \ldots \vee \bigcirc L^l_k)\\
&\wedge \Box(\neg p_1 \vee \Diamond p_2)\\
&\wedge \Box(\neg p_2 \vee \Diamond p_3)\\
&\vdots\\
&\wedge \Box(\neg p_n \vee \Diamond p_1),
\end{aligned}$$

where for each global clause, the literals $L^i_1, \ldots, L^i_k$ are generated by choosing k distinct variables randomly from the set $\{p_1, \ldots, p_n\}$ of n propositional variables and by determining the polarity of each literal with probability p. The sometime clauses included in φ only depend on the parameter n.

For the second class of PLTL-formulae, denoted by $\mathcal{C}^2_{ran}$, which will test the second part of the hypothesis, each randomly generated formula φ is a conjunction of initial, global and sometime clauses of the following form.

$$\begin{aligned}
&(r_1 \vee L^1_1 \vee \ldots \vee L^1_k) \wedge \ldots \wedge (r_1 \vee L^l_1 \vee \ldots \vee L^l_k)\\
&\wedge \Box(\neg r_n \vee \bigcirc r_1)\\
&\wedge \Box(\neg r_{n-1} \vee \bigcirc r_n)\\
&\vdots\\
&\wedge \Box(\neg r_1 \vee \bigcirc r_2)\\
&\wedge \Box(\neg r_n \vee \bigcirc \neg q_n) \wedge \ldots \wedge \Box(\neg r_1 \vee \bigcirc \neg q_n)\\
&\wedge (\neg r_1 \vee q_1) \wedge (\neg r_1 \vee \neg q_n)\\
&\wedge \Box(\neg q_1 \vee \Diamond s_2) \wedge \Box(\neg s_2 \vee q_2 \vee \bigcirc q_n \vee \ldots \vee \bigcirc q_3)\\
&\vdots\\
&\wedge \Box(\neg q_{n-1} \vee \Diamond s_n) \wedge \Box(\neg s_n \vee q_n)
\end{aligned}$$

For each of the first l initial clauses, the literals $L^i_1, \ldots, L^i_k$ are generated by choosing k distinct variables randomly from the set $\{p_1, \ldots, p_n\}$ of n propositional variables and by determining the polarity of each literal with probability p. The global and sometime clauses included in φ only depend on the parameter n.

In all our experiments, for both classes, the parameters k and p were fixed to 3 and 0.5, respectively.

The restricted form of global and sometime clauses in both classes indicates that they form strict subclasses of the class of sets (or conjunctions) of SNF clauses. Thus, not every PLTL-formula is equivalent to some formula in $\mathcal{C}^1_{ran}$or $\mathcal{C}^2_{ran}$. It turns out that the satisfiability problem of both $\mathcal{C}^1_{ran}$ and $\mathcal{C}^2_{ran}$ has reduced complexity.

Theorem 2. *The satisfiability problem for $\mathcal{C}^1_{ran}$ and $\mathcal{C}^2_{ran}$ is in* NP.

Proof sketch. Let φ be a SNF formula in $\mathcal{C}^1_{ran}$ over n propositional variables with global clauses $C_1, \ldots, C_l$. Let ψ' be the propositional formula $\text{next}(C_1) \wedge \ldots \wedge \text{next}(C_l)$. Let ψ_0 be the propositional formula $\psi' \wedge \neg p_0 \wedge \ldots \wedge \neg p_n$, and let ψ_i be the propositional formula $\psi' \wedge p_i$ for every i, $1 \leq i \leq n$. Then φ is satisfiable iff ψ_0 is satisfiable or for every i, ψ_i is satisfiable. Since each ψ_i, $0 \leq i \leq n$, is a propositional formula of length smaller than that of φ, these $n+1$ satisfiability test can be performed by a non-deterministic Turing machine in time polynomial in the length of φ.

Let φ be a SNF formula in $\mathcal{C}^2_{ran}$ over n propositional variables with global and sometime clauses $C_1, \ldots, C_{4n-2}$, and initial clauses $I_1, \ldots, I_{l+2}$. For an initial clause I_j let $I_j[r_1/\bot]$ denote the clause obtained from I_j by substituting $\bot$ (falsum) for r_1. It is straightforward to check that $G = r_1 \wedge \bigwedge_{1 \leq i \leq 4n-2} C_i$ is unsatisfiable. As far as the initial clauses are concerned, we see that almost all of them contain r_1 as a positive literal. Obviously, if $I = \bigwedge_{1 \leq i \leq l+2} I_i[r_1/\bot]$ is satisfiable, we can build a PLTL model $\mathcal{I}$ for φ where r_1 is false in the initial state s_0 and which satisfies all clauses of φ. However, if I is unsatisfiable, then we can only satisfy the set of initial clauses by assuming that r_1 is true in s_0. But then we will not be able to satisfy G. Thus, φ is unsatisfiable. So, φ is satisfiable iff I is satisfiable. Since I is a propositional formula of length smaller than φ, it follows that the satisfiability problem for $\mathcal{C}^2_{ran}$ is in NP. □

Underlying this theorem are considerations about the satisfiability equivalence of formulae and (un)satisfiability of particular subformulae. It should be stressed that the fact that a formula φ can be transformed into an 'easy' satisfiability equivalent formula φ' does not indicate that φ will be easy for a particular theorem prover. We can only do so if we know that a particular theorem prover actually performs the described transformations. However, the proof of Theorem 2 is the key to understanding the behaviour of the decision procedures under consideration.

To understand why $\mathcal{C}^1_{ran}$ and $\mathcal{C}^2_{ran}$ are suitable to test the first and second part of our hypothesis, respectively, we have to take a closer look at *TRP* and the

Initial resolution rules:
$$\frac{C_1 \vee L \qquad \sim L \vee C_2}{C_1 \vee C_2} \qquad\qquad \frac{C_1 \vee L \qquad \Box(\sim L \vee C_2)}{C_1 \vee C_2}$$

where $C_1 \vee L$ and $\sim L \vee C_2$ are initial clauses, $\Box(\sim L \vee C_2)$ is a global clause and $\sim L \vee C_2$ is a propositional clause.

Step resolution rules:
$$\frac{\Box(C_1 \vee L) \qquad \Box(\sim L \vee C_2)}{\Box(C_1 \vee C_2)}$$

$$\frac{\Box(C_1 \vee L) \qquad \Box(\bigcirc \sim L \vee D_2)}{\Box(C_1 \vee D_2)} \qquad \frac{\Box(D_1 \vee \bigcirc L) \qquad \Box(\bigcirc \sim L \vee D_2)}{\Box(D_1 \vee D_2)}$$

where $\Box(C_1 \vee L)$ and $\Box(\sim L \vee C_2)$ are global clauses and $C_1 \vee L$ and $\sim L \vee C_2$ are propositional clauses, and $\Box(\bigcirc \sim L \vee D_2)$ and $\Box(D_1 \vee \bigcirc L)$ are global clauses.

Temporal resolution rule:
$$\frac{\begin{array}{cc}\Box(C^1_1 \vee \bigvee_{l=1}^{k^1_1} \bigcirc D^1_{1,l}) & \Box(C^n_1 \vee \bigvee_{l=1}^{k^n_1} \bigcirc D^n_{1,l}) \\ \vdots & \vdots \\ \Box(C^1_{m_1} \vee \bigvee_{l=1}^{k^1_{m_1}} \bigcirc D^1_{m_1,l}) \cdots & \Box(C^n_{m_n} \vee \bigvee_{l=1}^{k^n_{m_n}} \bigcirc D^n_{m_n,l}) \quad \Box(C \vee \Diamond L)\end{array}}{\Box(C \vee (\sim(\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} C^i_j) \mathcal{W} L))}$$

where for all i, $1 \leq i \leq n$, $(\bigwedge_{j=1}^{m_i} \bigvee_{l=1}^{k^i_j} D^i_{j,l}) \rightarrow \sim L$ and $(\bigwedge_{j=1}^{m_i} \bigvee_{l=1}^{k^i_j} D^i_{j,l}) \rightarrow (\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} C^i_j)$ are provable.

Figure 1: The temporal resolution calculus

temporal resolution calculus. Besides the standard resolution rule for propositional clause logic (which can only be applied to initial clauses), the temporal resolution calculus contains two inference rules, *step resolution* and *temporal resolution*; see Figure 1. While the step resolution rule resolves two global clauses C_1 and C_2 on complementary literals in $\text{next}(C_1)$ and $\text{next}(C_2)$, the temporal resolution rule resolves a set of global clauses with a sometime clause to derive a set of initial and global clauses. Determining a set of global clauses which can be successfully resolved with a sometime clause is a non-trivial problem, and our implementation follows the approach described in [6]. In *TRP* propositional and step resolution take precedence over temporal resolution. That is, *TRP* will only try to apply the temporal resolution rule when no further applications of the other two rules are possible.

The class $\mathcal{C}^1_{ran}$ has been constructed in such a way that for an unsatisfiable formula φ in $\mathcal{C}^1_{ran}$ either the conjunction of global clauses in φ is already unsatisfiable, which can be detected without any application of the temporal resolution rule, or a small number of comparatively easy applications of the temporal resolution rule will allow us to derive a contradiction. Basically, this is due to the uniform character of the global clauses in φ which impose the same constraints on all states in a potential PLTL model.

In contrast, the tableaux-based decision procedures will try to construct a model for φ. Let $C_1, \ldots, C_l$ be the global clauses in φ. First note, that $C_i = \Box(\bigcirc L^i_1 \vee \bigcirc L^i_2 \vee \bigcirc L^i_3)$ is equivalent to $\bigcirc \Box(L^i_1 \vee L^i_2 \vee L^i_3)$. Thus, C_i puts no constraint on the state s_0 in a potential model $\mathcal{I}$ for φ, and enforces the constraint $L^i_1 \vee L^i_2 \vee L^i_3 = \text{next}(C_i)$ on the remaining states s_j, $j > 0$, of $\mathcal{I}$. In analogy to *TRP*, a tableaux-based procedure may easily detect that no model for φ exists if $C_\Box = \bigwedge_{1 \leq i \leq l} \text{next}(C_i)$ is unsatisfiable. This will depend on the presence of sufficiently strong simplification methods for Boolean formulae in the decision procedure. However, if $C_\Box$ is satisfiable, the tableaux-based procedure will construct a set or sequence of states, each with its own valuation for the propositional variables $p_1, \ldots, p_n$, and each satisfying $C_\Box$. In the worst case the procedure constructs 2^n distinct states before this part of the model construction is finished. Note that according to the proof of Theorem 2 a model with at most n states exists for each formula in $\mathcal{C}^1_{ran}$. The procedures will only detect that the interpretation they have constructed is not a model for φ once they have checked whether all the sometime formulae $\Diamond p_i$ contained in the sometime clauses of φ can be satisfied (this is called the *eventuality check*). In the case of *STeP*, this is done in a separate phase, which follows the model construction phase. In this model construction phase *STeP* effectively builds a representation of not just one, but all models of the formula under consideration. This means *STeP* should be very sensitive to the total number of models and consequently we expect it to be the worst performing system on $\mathcal{C}^1_{ran}$. In the case of Schwendimann's one-phase tableau calculus only one interpretation is constructed

at a time and the eventuality check is performed during the search for a model, which should result in better computational behaviour as compared to *STeP*. However, the Schwendimann algorithm will then backtrack and search for an alternative interpretation satisfying φ and only after all alternatives are exhausted will the algorithm conclude that φ is unsatisfiable.

Now consider a satisfiable formula φ in $\mathcal{C}^2_{ran}$ over n propositional variables with global and sometime clauses $C_1, \ldots, C_{4n-2}$, and initial clauses $I_1, \ldots, I_{l+2}$. According to the proof of Theorem 2, φ can only be satisfiable if $I = \bigwedge_{1 \leq i \leq l+2} I_i[r_1/\bot]$ is satisfiable. The initial clauses are the first part of φ considered by any of the tableaux-based decision procedures. Assuming that r_1 is set to false in the initial state, it will be an easy task for the procedure to find a satisfying valuation for the propositional variables $p_1, \ldots, p_n$ occurring in I, the remaining clauses of φ can be satisfying by setting all s_i and q_i, $1 \leq i \leq n$ to false for all states including the initial state. Thus, assuming that the tableaux-based decision procedure searches for a model in the way outlined, it will easily find the model we have just described. The temporal resolution procedure *TRP* on the other hand will always consider the global and sometime clauses of φ regardless as to whether r_1 can be assumed to be false or not. The global and sometime clauses have been chosen in such a way that a successful application of the temporal resolution is possible to each of the sometime clauses, however, only if performed in the correct ordering, starting with $\Box(\neg q_{n-1} \vee \Diamond s_n)$ and ending with $\Box(\neg q_1 \vee s_2)$. For successful and unsuccessful applications of the temporal resolution rule, the global clauses make sure that each application is non-trivial and takes considerable time to perform. After completing all successful applications of the temporal resolution rule, *TRP* will derive $\Box\neg r_1$, which in turn can be used to remove all positive occurrences of r_1 from the initial clauses. However, the simplified initial clauses remain satisfiable. Thus considerable computational resources have been invested into the exploration of a part of φ which a tableaux-based procedure was able to ignore.

What remains to be done, is to verify by empirical experiment that this theoretical analysis correctly reflects the actual behaviour of the procedures under consideration.

4 Experiments

We will now see how *TRP*, *STeP*, and the two tableau calculi incorporated in the Logics Workbench 1.1 perform on $\mathcal{C}^1_{ran}$ and $\mathcal{C}^2_{ran}$.

The tests have been performed on a PC with a 1.3GHz AMD Athlon processor, 512MB main memory, and 1GB virtual memory running RedHat Linux 7.0. For each individual satisfiability test a time-limit of 1000 CPU seconds was used.

Concerning the satisfiability of random formulae in $\mathcal{C}^1_{ran}$ and $\mathcal{C}^2_{ran}$ we observe that similar to random kSAT formulae, a random SNF formula is likely to be satisfiable if the number l of clauses is small and it is likely to be unsatisfiable if the number l of clauses is large. Figure 2 shows the percentage of satisfiable random formulae in both classes for $n = 5$ and $n = 12$. To obtain these graphs, for each of the values of l between 1 and $8n$ we have generated a test set of 100 random SNF formulae which were tested for satisfiability. In the case of $\mathcal{C}^1_{ran}$ we see that for ratios l/n smaller than 2, a random formula is most likely satisfiable while for ratios l/n greater than 4, it is most likely unsatisfiable. For ratios l/n between 2 and 4 we note a phase transition in the satisfiability of random SNF formula which becomes more abrupt with greater values of n. For $\mathcal{C}^2_{ran}$, a random formula is most likely satisfiable for ratios l/n smaller than 3.5 and most likely unsatisfiable for ratios l/n greater than 6.5. These graphs have been obtained using the reduction of the satisfiability problem of $\mathcal{C}^1_{ran}$ and $\mathcal{C}^2_{ran}$ to propositional logic described in the proof of Theorem 2. The tests have been performed using the propositional logic module of the Logics Workbench 1.1. None of test required more than 80 milliseconds.

Figure 3 shows the median CPU time graphs of all four procedures on $\mathcal{C}^1_{ran}$ and $\mathcal{C}^2_{ran}$ for $n = 5$ and $n = 12$. In each graph the vertical line indicates the ratio l/n at which test sets contain 50% satisfiable and 50% unsatisfiable formulae. We observe that on $\mathcal{C}^1_{ran}$, *TRP* performs better than any of the tableaux-based decision procedures except for certain ratios where it is outperformed by Janssen's algorithm (the 'satisfiable' function of the Logics Workbench 1.1).

On $\mathcal{C}^1_{ran}$ for $n = 12$, Janssen's algorithm behaves roughly as predicted in Section 3. If φ is a formula in $\mathcal{C}^1_{ran}$ over n propositional variables with global clauses $C_1, \ldots, C_l$ where $C_i = \Box(\bigcirc L^i_1 \vee \bigcirc L^i_2 \vee \bigcirc L^i_3)$, then each C_i enforces the constraint $L^i_1 \vee L^i_2 \vee L^i_3 = \text{next}(C_i)$ on the states s_j, $j > 0$, of any interpretation $\mathcal{I}$ which may satisfy φ. With increasing l, the number of distinct valuations which satisfy $\text{next}(C_i)$ for all i, $1 \leq i \leq l$, continuously decreases. For $l/n > 5$, no such satisfying valuation exists for more than 50% of the formulae in $\mathcal{C}^1_{ran}$. This is easy to detect and leads to the good performance of Janssen's algorithm on formulae in this region for $n = 12$ on $\mathcal{C}^1_{ran}$. For

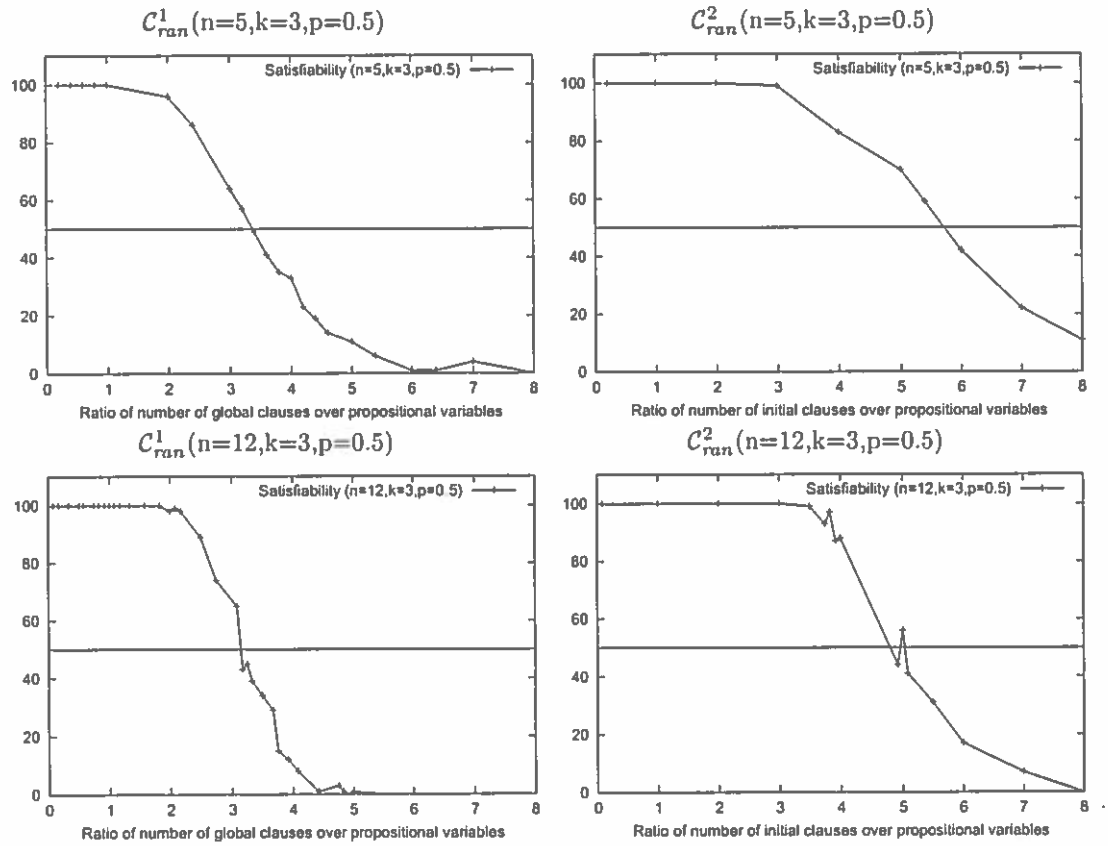


Figure 2: Satisfiability of random PLTL-formulae

all ratios l/n smaller than 5 the sometime clauses in φ have to be taken into account. Consider an arbitrary sometime clause $\Box(\neg p_i \vee \Diamond p_{i+1})$. This clause imposes the constraint $\neg p_i \vee \Diamond p_{i+1}$ on all states in $\mathcal{I}$. To satisfy this disjunction, for s_0, the implementation of Janssen's algorithm in the Logics Workbench 1.1 will always choose to make $\neg p_i$ true, but for all other states this will only be possible if p_i is not already true due to the constraints imposed by the global clauses. Thus, as long as there is a satisfying valuation which makes all next(C_i) true and also all p_i false, $\mathcal{I}$ can be constructed easily. For $n = 12$ we find that for $l/n \leq 0.33$ this is true for more than 50% of the formulae and for $l/n \leq 0.58$ it is still true for more than 40% of the formulae. The percentile graph for $n = 12$ shows quite a good correlation between the sudden rise in the CPU time consumption and the decline of the percentage of formulae for which models can be found in the way described above.

To explain the behaviour of the Logics Workbench for $n = 12$ and ratios $0.83 \leq l/n \leq 5$ we take Theorem 2 into account. The proof of Theorem 2 tells us something about the structure of PLTL-models for random SNF formulae. There are basically two possibilities. Either the model consists of an infinite sequence of states whose identical valuations satisfy the formula ψ_0 defined in the proof of Theorem 2, or the model consists of an infinite repetition of the sequence of states $s_1, \ldots, s_n$ where each s_i satisfies the formulae ψ_i defined above. The first possibility provides an explanation of the good behaviour of Janssen's algorithm for ratios $l/n \leq 0.58$. However, for ratios $0.83 \leq l/n \leq 5$ almost all models will be structured according to the second possibility. Since the the implementation of Janssen's algorithm in the Logics Workbench 1.1. is not aware that the states $s_1, \ldots, s_n$ have to satisfy particular valuations, it searches through up to n^{2^n} combinations of possible valuations, until a model has been found. Since the global clauses put ever stronger constraints on satisfying constraints with increasing ratio l/n, the search space is getting continuously smaller. Thus, the continuous decrease in the median CPU time observed after the peak at $l = 1.17$. Curiously, this peak occurs at a ratio l/n which is not related to the ratio $l/n = 3.15$ where 50% of the formula are satisfiable and 50% of the formulae are unsatisfiable.

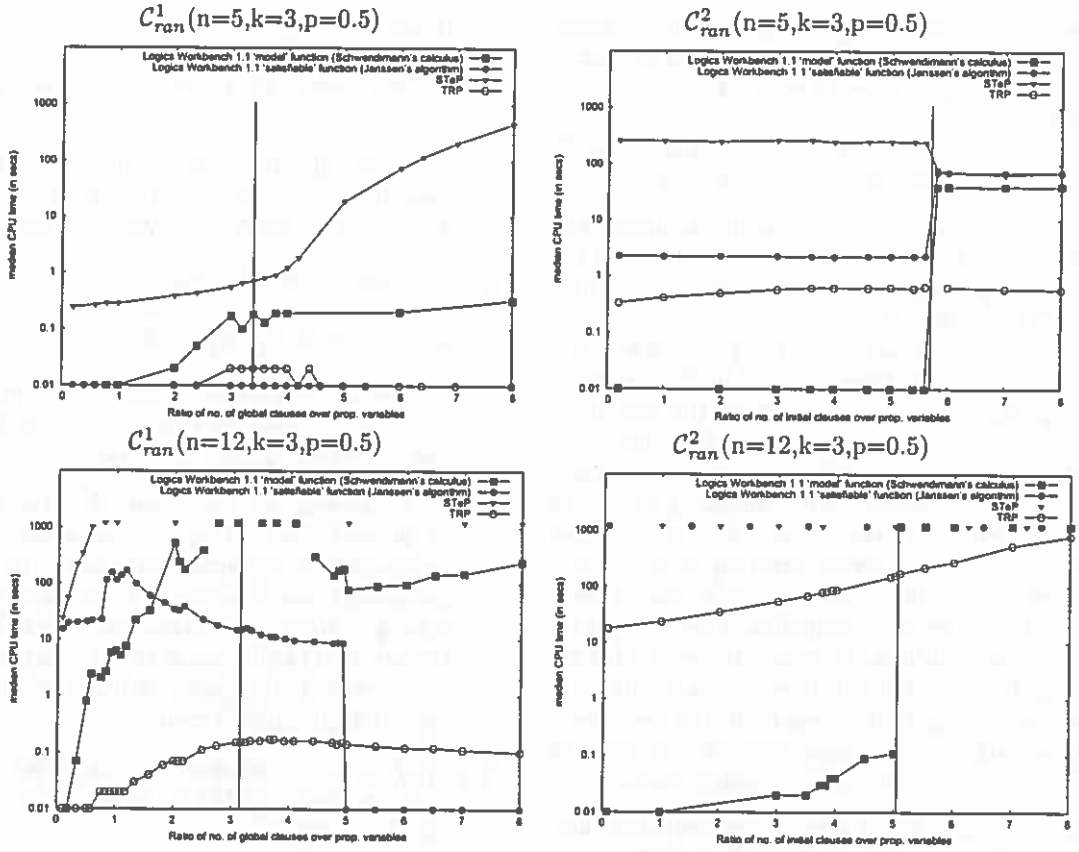


Figure 3: Performance of the decision procedures

As predicted, the performance of *STeP* is worse than that of the Logics Workbench 1.1 'satisfiable' function. We have already observed that on $\mathcal{C}^1_{ran}$ for $n = 12$ and $l/n < 0.58$ the majority of the formulae under consideration are satisfied by an interpretation consisting of an infinite sequence of states whose identical valuations satisfy the formula ψ_0 defined in the proof of Theorem 2. Only for these formulae *STeP* terminates within the imposed time-limit of 1000 CPU seconds. For all other formulae the simultaneous consideration of all potentially satisfying interpretations in conjunction with a delayed eventuality check provides the explanation for the inferior performance of *STeP*. A surprise, however, is the behaviour of *STeP* on $\mathcal{C}^1_{ran}$ for $n = 5$ and $l/n > 4.5$ where the median CPU time increases. Since the number of potentially satisfying interpretations that need to be considered decreases in this area, we were expecting a much better performance of *STeP*, similar to the good performance we see for the Logics Workbench 1.1 'satisfiable' function for $n = 12$ and $l/n > 5$.

The Logics Worbench 1.1 'model' function behaves uniformly better than *STeP* and is also able to outperform the Logics Workbench 'satisfiable' function on $\mathcal{C}^1_{ran}$ for $n = 12$ and $l/n < 1.8$. The behaviour of Schwendimann's one-phase tableau calculus is not so much influenced by the total number of models that the formula under consideration has. Instead it is influenced by the number of backtracks that the system needs to perform until a model has been found. This in turn is influenced by the simplification and redundancy elimination rules and by the heuristics which determine the sequence of tableau expansion steps the system performs which all may allow the system to avoid the introduction of superfluous backtrack points. Unfortunately, the implementation of Schwendimann's calculus lacks any simplification and redundancy elimination rules and does not use sophisticated heuristics to guide its search. This becomes evident on $\mathcal{C}^1_{ran}$ for $n = 12$ and $l/n \geq 5$, where the Logics Worbench 1.1 'model' function should be much faster in determining that the formulae in this region are unsatisfiable.

TRP behaves as predicted. As pointed out above, for $n = 12$ and $l/n > 5$, the conjunction of global clauses in a formula of $\mathcal{C}^1_{ran}$ is unsatisfiable. The slow and steady increase in the time needed to decide $\mathcal{C}^1_{ran}$ for-

mulae for $l/n \geq 5$ is due to the increase in the number of global clauses in the formulae which are saturated under the three inference rules. For $n = 12$ and $l/n < 0.83$ the number of resolvents generated is below 10, reaching about 1000 resolvents for $l/n = 5$, and dropping to below 350 resolvents for $l/n > 6$.

For C^2_{ran} we see that *TRP* is orders of magnitude slower than the 'model' function of the Logics Workbench 1.1 on all satisfiable formulae generated for $n = 5$ and $n = 12$. This is due to the fact that this algorithm behaves exactly as described in the previous section, given that the clauses of the formulae in C^2_{ran} are written in a particular order which allows the algorithm to pick the clauses and literals of the formulae in an optimal order. Although this shows that the considerations in the previous section concerning these two decision procedures are correct, it is interesting to see that for $n = 12$ the 'model' function of the Logics Workbench is not able to solve a single unsatisfiable formula in C^2_{ran}. Concerning the Logics Workbench 1.1 'satisfiable' function it is important to point out that for $n = 12$ the algorithm fails to solve a single problem not because it exceeds the time-limit, but because it terminates with a segmentation fault after exhausting the space in one of its internal data structures.

Summing up, for the first class of problems the characteristics of C^1_{ran} roughly leads to the behaviour predicted in Section 3. For the second class of problems the predicted behaviour is observed only when comparing the curves for *TRP* and the 'model' function implementation of Schwendimann's calculus. The behaviour of the other tableaux implementations is not in accordance with the hypothesis. This indicates that further analysis is necessary before the hypothesis can be regarded as empirically validated. However, the results of the experiments are evidence of a correlation between the structural properties of the problems and the computational behaviour of the implemented decision procedures for PLTL.

References

[1] F. Baader, B. Hollunder, B. Nebel, H.-J. Profitlich, and E. Franconi. An empirical analysis of optimization techniques for terminological representation systems or "Making KRIS get a move on". *Applied Intelligence*, 4(2):109–132, May 1994.

[2] F. Baader and S. Tobies. The inverse method implements the automata approach for modal satisfiability. In *Proc. IJCAR 2001*, volume 2083 of *LNAI*, pages 92–106. Springer, 2001.

[3] M. Cadoli, A. Giovanardi, and M. Schaerf. An algorithm to evaluate quantified boolean formulae. In *Proc. AAAI-98*, pages 262–267. AAAI Press, 1998.

[4] H. De Nivelle, R. A. Schmidt, and U. Hustadt. Resolution-based methods for modal logics. *Logic Journal of the IGPL*, 8(3):265–292, May 2000.

[5] C. Dixon. Search strategies for resolution in temporal logics. In *Proc. CADE-13*, LNAI 1104, pages 673–687. Springer, 1996.

[6] C. Dixon. Using Otter for temporal resolution. In *Advances in Temporal Logic*, volume 16 of *Applied Logic Series*, pages 149–166. Kluwer, 2000.

[7] F. M. Donini, P. Liberatore, F. Massacci, and M. Schaerf. Generating symbolic model-checking benchmarks by reduction from QBF. In *Issues in the Design and Experimental Evaluation of Systems for Modal and Temporal Logics*, Technical Report DII 14/01, pages 10–18, Italy, 2001. Dipartimento di Ingegneria dell'Informazione, Universitá degli Studi di Siena.

[8] M. Fisher. A resolution method for temporal logic. In *Proc. IJCAI'91*, pages 99–104. Morgan Kaufman, 1991.

[9] M. Fisher, C. Dixon, and M. Peim. Clausal temporal resolution. *ACM Transactions on Computational Logic*, 2(1):12–56, 2001.

[10] J. Franco and M. Paull. Probabilistic analysis of the Davis Putnam procedure for solving the satisfiability problem. *Discrete Applied Mathematics*, 5:77–87, 1983.

[11] I. P. Gent and T. Walsh. Towards an understanding of hill-climbing procedures for SAT. In *Proc. AAAI-93*, pages 28–33. AAAI Press, 1993.

[12] A. Goldberg, P. Purdom, and C. Brown. Average time analyses of simplified Davis-Putnam procedures. *Information Processing Letters*, 15(2):72–75, 1982.

[13] A. Goldberg, P. Purdom, and C. Brown. Corrigendum: "Average time analyses of simplified Davis-Putnam procedures", Information Processing Letters 15(2):72–75. *Information Processing Letters*, 16(4):213–213, 1983.

[14] G. D. Gough. Decision procedures for temporal logic. Technical Report UMCS-89-10-1, Department of Computer Science, University of Manchester, UK, 1989.

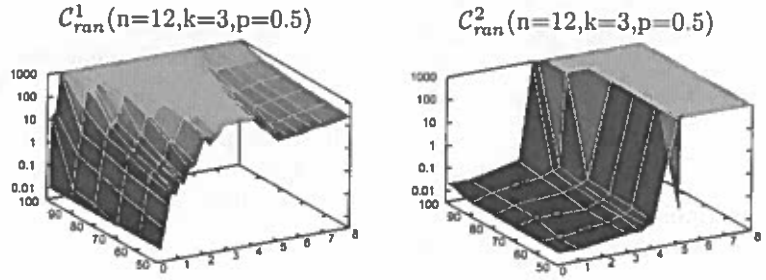


Figure 4: CPU time percentiles for the 'model' function of the Logics Workbench 1.1 (Schwendimann's calculus)

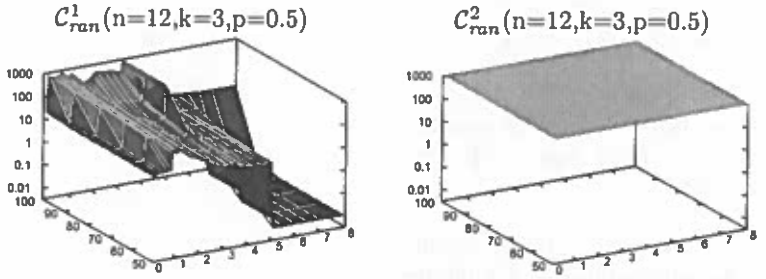


Figure 5: CPU time percentiles for the 'satisfiable' function of the Logics Workbench 1.1 (Janssen's algorithm)

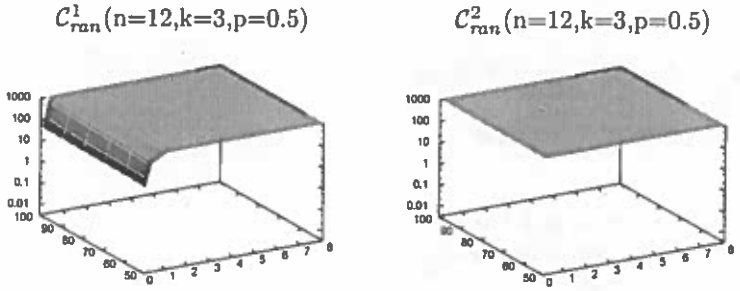


Figure 6: CPU time percentiles for *STeP*

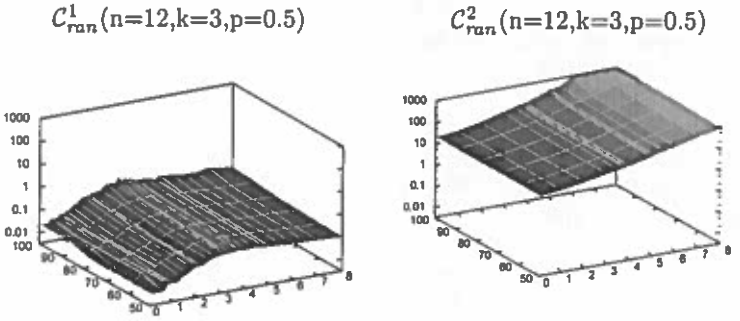


Figure 7: CPU time percentiles for *TRP*

[15] J. N. Hooker. Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1:33–42, 1996.

[16] I. Horrocks and P. F. Patel-Schneider. Evaluating optimised decision procedures for propositional modal $\mathbf{K}_{(\mathbf{m})}$ satisfiability. In *SAT2000: Highlights of Satisfiability Research in the Year 2000*. ISO Press, 2000.

[17] I. Horrocks, P. F. Patel-Schneider, and R. Sebastiani. An analysis of empirical testing for modal decision procedures. *Logic Journal of the IGPL*, 8(3):293–323, 2000.

[18] U. Hustadt and R. A. Schmidt. An empirical analysis of modal theorem provers. *Journal of Applied Non-Classical Logics*, 9(4):479–522, 1999.

[19] U. Hustadt and R. A. Schmidt. On the relation of resolution and tableaux proof systems for description logics. In *Proc. IJCAI'99*, pages 110–115. Morgan Kaufmann, 1999.

[20] U. Hustadt and R. A. Schmidt. Using resolution for testing modal satisfiability and building models. In *SAT2000: Highlights of Satisfiability Research in the Year 2000*, pages 459–483. IOS Press, 2000.

[21] G. Janssen. *Logics for Digital Circuit Verification: Theory, Algorithms, and Applications*. PhD thesis, Eindhoven University of Technology, The Netherlands, 1999.

[22] Y. Kesten, Z. Manna, H. McGuire, and A. Pnueli. A decision algorithm for full propositional temporal logic. In *Proc. CAV'93*, volume 697 of *LNCS*, pages 97–109. Springer, 1993.

[23] Z. Manna and the STeP group. STeP: The Stanford Temporal Prover. Technical report STAN-CS-TR-94-1518, Department of Computer Science, Stanford University, USA, 1994.

[24] P. F. Patel-Schneider and R. Sebastiani. A system and methodology for generating random modal formulae. In *Proc. IJCAR 2001*, volume 2083 of *LNAI*, pages 464–468. Springer, 2001.

[25] J. Rintanen. Improvements to the evaluation of quantified booelan formulae. In *Proc. IJCAI'99*, pages 1192–1197. Morgan Kaufmann, 1999.

[26] S. Schwendimann. *Aspects of Computational Logic*. PhD thesis, Universität Bern, Switzerland, 1998.

[27] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proc. AAAI-92*, pages 440–446. MIT Press, 1992.

[28] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logic. *Journal of the ACM*, 32(3):733–749, 1985.

[29] P. Wolper. The tableau method for temporal logic: An overview. *Logique et Analyse*, 28:119–136, 1985.

Complexity Issues III

As Time Goes By: Automatic Complexity Analysis of Simplification Rules

Thom Frühwirth
Ludwig-Maximilians-Universität München
Oettingenstrasse 67, D-80538 Munich, Germany
www.informatik.uni-muenchen.de/~fruehwir/

Abstract

From a suitable termination order, called a tight ranking, we can automatically compute the worst-case time complexity of a CHR constraint simplification rule program from its program text: We combine the worst-case derivation length of a query predicted from its ranking with a worst-case estimate of the number and cost of rule application attempts and the cost of rule applications to obtain the desired meta-theorem. For two Boolean and a path consistency solver, the predictions are computed and are compared to some empirical run-time measurements.

1 Introduction

The programming language *CHR (Constraint Handling Rules)* [Fru98] was originally introduced to ease the development of constraint solvers. Currently several CHR libraries exist in languages such as Prolog, Haskell and Java, and dozens of projects use CHR. Over time it has become apparent that CHR and its extensions [Abd00] are useful for implementing reasoning systems in general, including deduction and abduction, since techniques like forward and backward chaining, bottom-up and top-down evaluation, integrity constraints, tabulation/memoization can be easily implemented and combined [AAI00, SaAb00].

CHR are a committed-choice concurrent constraint logic programming language consisting of guarded rules that work on conjunctions of constraints. A CHR program consists of simplification and propagation rules. Simplification replaces constraints by simpler constraints while preserving logical equivalence. Propagation adds new constraints which are logically redundant but may cause further simplification.

Properties like rule confluence [AFM99] and program equivalence [AbFr99] have been investigated for CHR. These properties are decidable for terminating programs. In a previous paper [Fru00a] we have proven termination of simplification rule programs using rankings. A ranking maps lhs (left hand side) and rhs (right hand side) of each simplification rule to a natural number, such that the rank of the lhs is strictly larger than the rank of the rhs. A given constraint satisfaction problem is posed as a query to the CHR solver. Intuitively then, the rank of a query yields an upper bound on the number of rule applications (derivation steps), i.e. derivation lengths [Fru00b], because each rule application decreases the rank by at least one.

Example 1.1 *Consider the constraint* even *that ensures that a natural number (written in successor notation) is even:*

```
even(0) <=> true.
even(s(N)) <=> N=s(M), even(M).
```

The first rule says that even(0) *can be simplified to* true*, a built-in constraint that is always satisfiable. In the second rule, the built-in constraint* = *stands for syntactic equality:* N=s(M) *ensures that* N *is the successor of some number* M*. The comma stands for conjunction* ∧*. The rule says that if the argument of* even *is the successor of some number* N*, then the predecessor of this number,* M*, must be even.*

If a constraint matches the lhs of a rule, it is replaced by the rhs of the rule. If no rule matches a constraint, the constraint delays. For example, the query even(N) *delays. When the constraint* N=0 *is added,* even(N) *is woken and behaves like the query* even(0)*. It reduces to* true *with the first rule. To the query* even(s(X)) *the second rule is applicable, the answer is* X=s(M),even(M)*. The query* even(s(0)) *results in an inconsistency after application of the second rule, since* 0=s(M) *is unsatisfiable.*

An obvious ranking for the rules of even *is*

$$rank(even(N)) = size(N)$$

$$size(0) = 1$$
$$size(s(N)) = 1 + size(N)$$

The ranking not only proves termination, it also gives an upper bound on the derivation length, in case the argument of even *is completely known (ground): With each rule application, we decrease the rank of the argument of* even *by* 2.

In [Fru00b] we have shown that the derivation length is not a suitable measure for time complexity. The runtime of a CHR program not only depends on the number of rule applications, but also, more significantly, on the number of rule application *attempts* (rule tries).

In this paper we combine the predicted worst-case derivation length with a worst-case estimate of the number and cost of rule tries and the cost of rule applications to obtain a meta-theorem for the worst-case time complexity of CHR constraint simplification rule programs. In the theorem, we will make no specific assumptions on the implementation of CHR, so it applies to naive implementations of CHR as well.

Example 1.2 *[Contd.] It is easy to show that the worst-case time complexity of a single* even *constraint is linear in the derivation length, i.e. the rank. The same observation holds for a query consisting of several ground* even *constraints, if the rank is defined as the sum of the ranks of the individual constraints.*

However, things change when we add the rule:

```
even(s(X)),even(X) <=> false.
```

where false *is a built-in constraint that is always unsatisfiable. This rule may be applicable to* all pairs of even *constraints in a query, and again after a reduction of a single* even *constraint with one of the other two rules. Of course in most cases, the rule application attempts (rule tries) will be in vain.*

Thus the number of rule tries in a single derivation step is quadratic in the number of even *constraints in the query. Since the rank of an* even *constraint is at least one, the rank of the query is a bound on the number of constraints. The number of derivation steps is also bounded by the rank of the query. Overall, this yields an algorithm that is cubic in the rank of the query.*

Related Work. To the best of our knowledge, only [McA99, GaMc01] are closely related to our work in that they give several complexity meta-theorems for a logical rule-based language. These papers investigate bottom-up logic programming as a formalism for expressing static analyses and related algorithms. [McA99] is concerned with certain propagation rules (in our terminology), while [GaMc01] extends the rule language with deletions of atomic formulae and static priorities between rules. Such rules correspond to CHR simplification or simpagation rules [Fru98] that are applied in textual order.

McAllester and Ganzinger prove several complexity theorems which allow, in many cases, to determine the asymptotic running time of a bottom-up logic programm by inspection. The main difference and complementarity between their work and our paper is that they consider rules that must be applied to ground formulae at run-time, while we consider simplification rules that involve free variables at run-time and arbitrary built-in constraints. They deal with the complexity of optimally implemented programs using clever indexing and structure sharing, while our results apply also to naive implementations of CHR.

In the complexity meta-theorem of [GaMc01], the complexity is the sum of the syntactic size of the query and the worst number of potential prefix firings (ground sub-formulas of lhs instances that could occur in a derivation) of the rules in the program. Here it is - ignoring the cost of built-in constraints - the sum of the rank of the query and the number of potential rule applications. The computation of the number of prefix firings requires insights about the states of all valid computations that can be performed. The number of potential rule applications can be computed automatically from the program text, once a ranking is known.

Overview of the Paper. We will first give syntax and semantics of CHR. In Section 3, we introduce rankings and show how they can be used to derive tight upper bounds for worst-case derivation lengths. In the next section we show how to use these derivation lengths to predict the worst-case complexity of CHR programs. Finally, the fifth section reviews some CHR constraint solver programs. Based on the predicted worst-case derivation lengths, the worst-case time complexity is computed according to our complexity meta-theorem. The prediction is compared with empirical run-time measurements. We conclude with a discussion of the results obtained. This paper is a revised version of a paper presented at a non-archival workshop [Fru01]. We apply our complexity theorem to two Boolean constraint solvers and one path con-

sistency solver. Preliminary empirical results indicate that the actual run-time of the solvers in the Sicstus Prolog implementation of CHR is better due to CHR compiler optimizations like indexing.

2 Syntax and Semantics

In this section we give syntax and semantics for CHR, for details see [AFM99]. We assume some familiarity with (concurrent) constraint (logic) programming [JaMa94, MaSt98].

A *constraint* is a predicate (atomic formula) in first-order logic. We distinguish between *built-in (or predefined) constraints* and *CHR (or user-defined) constraints*. Built-in constraints are those handled by a given constraint solver. CHR constraints are those defined by a CHR program.

In the following definitions, upper case letters stand for conjunctions of constraints.

Definition 2.1 *A CHR program is a finite set of CHR. There are two kinds of CHR. A* simplification CHR *is of the form*

$$n \ @ \ H \ \texttt{<=>} \ G \mid B$$

and a propagation CHR *is of the form*

$$n \ @ \ H \ \texttt{==>} \ G \mid B$$

where the rule has an optional name n *followed by the symbol @. The lhs H (head) is a conjunction of CHR constraints. The optional guard G followed by the symbol | is a conjunction of built-in constraints. The rhs B (body) is a conjunction of built-in and CHR constraints.*

The *operational semantics* of CHR programs is given by a state transition system. With *derivation steps (transitions, reductions)* one can proceed from one state to the next.

Definition 2.2 *A* state (or: goal) *is a conjunction of built-in and CHR constraints. An* initial state (or: query) *is an arbitrary state. In a* final state (or: answer) *either the built-in constraints are inconsistent or no derivation step is possible anymore. A* derivation *is a sequence of derivation steps* $S_1 \longmapsto S_2 \longmapsto S_3 \ldots$. *The* derivation length *is the number of derivation steps in a derivation.*

Definition 2.3 *Let P be a CHR program and CT be a constraint theory for the built-in constraints. The transition relation* $\longmapsto$ *for CHR is as follows:*

Simplify
$H' \wedge C \longmapsto (H = H') \wedge G \wedge B \wedge C$
if $(H \ \texttt{<=>} \ G \mid B)$ *in* P *and* $CT \models C \rightarrow \exists \bar{x}(H = H' \wedge G)$

Propagate
$H' \wedge C \longmapsto (H = H') \wedge G \wedge B \wedge H' \wedge C$
if $(H \ \texttt{==>} \ G \mid B)$ *in* P *and* $CT \models C \rightarrow \exists \bar{x}(H = H' \wedge G)$

When we use a rule from the program, we will rename its variables using new symbols, and these variables are denoted by the sequence $\bar{x}$. A rule with lhs H and guard G is *applicable* to CHR constraints H' in the context of constraints C, when the condition holds that $CT \models C \rightarrow \exists \bar{x}(H = H' \wedge G)$.

Any of the applicable rules can be applied, but it is a committed choice, it cannot be undone. If an applicable simplification rule $(H \ \texttt{<=>} \ G \mid B)$ is applied to the CHR constraints H', the **Simplify** transition removes H' from the state, adds the rhs B to the state and also adds the equation $H = H'$ and the guard G. If a propagation rule $(H \ \texttt{==>} \ G \mid B)$ is applied to H', the **Propagate** transition adds B, $H = H'$ and G, but does not remove H'. Trivial non-termination is avoided by applying a propagation rule at most once to the same constraints [Abd97].

We finally discuss in more detail *the rule applicability condition* $CT \models C \rightarrow \exists \bar{x}(H = H' \wedge G)$. The equation $(H = H')$ is a notational shorthand for equating the arguments of the CHR constraints that occur in H and H'. More precisely, by $(H_1 \wedge \ldots \wedge H_n) = (H'_1 \wedge \ldots \wedge H'_n)$, where conjuncts can be permuted, we mean $(H_1 = H'_1) \wedge \ldots \wedge (H_n = H'_n)$. By equating two constraints, $c(t_1, \ldots, t_n) = c(s_1, \ldots, s_n)$, we mean $(t_1 = s_1) \wedge \ldots \wedge (t_n = s_n)$. The symbol $=$ is to be understood as built-in constraint for syntactic equality.

Operationally, the rule applicability condition can be checked as follows: Given the built-in constraints of C, try to solve the built-in constraints $(H = H' \wedge G)$ without further constraining any variable in H' and C. This means that we first check that H' matches H and then check the guard G under this matching.

As a consequence, in a CHR implementation, there are several computational phases when a rule is applied:

LHS Matching: Atomic CHR constraints in the current state have to be found that match the lhs constraints of the rule.

Guard Checking: It has to be checked if the current built-in constraints imply the guard of the rule.

RHS Handling: The built-in and CHR constraints of the rhs are added. Before that, the CHR constraints of the lhs are removed.

In this paper we are only concerned with simplification rules. For the rest of the paper we assume that CHR programs do not contain any propagation rules.

3 Rankings for Derivation Lengths

In this section, we introduce rankings for constraint simplification rules and show how the rankings can be used to derive tight upper bounds for worst-case derivation lengths of CHR programs.

3.1 Rankings

A ranking maps terms and formulae to natural numbers.

Definition 3.1 *Let f be a function or predicate symbol of arity $n(n \geq 0)$ and let $t_i(1 \leq i \leq n)$ be terms. A* (linear polynomial) CHR ranking (function) *defines the rank of a term or constraint atom $f(t_1, \ldots, t_n)$ as a natural number:*

$$rank(f(t_1, \ldots, t_n)) = a_0^f + a_1^f * rank(t_1) + \ldots + a_n^f * rank(t_n)$$

where the a_i^f are natural numbers. For each variable X we impose $rank(X) \geq 0$. For each built-in constraint C we impose $rank(C) = 0$. For each CHR constraint D we impose $rank(D) > 0$.

The rank of a conjunction is the sum of the ranks of its conjuncts:

$$rank((A \wedge B)) = rank(A) + rank(B)$$

The rank of any built-in constraint is 0, since we assume that they always terminate and since their derivation length is 0. A built-in constraint may imply order constraints between the ranks of its arguments (interargument relations), such as $s = t \rightarrow rank(s) = rank(t)$.

In [Fru00a] we prove termination for CHR simplification rule programs under any scheduling of rule applications using CHR rankings. We want to make sure that the rank of the lhs of a rule is always strictly larger than the rank of the rhs of the rule.

Definition 3.2 *Let rank be a CHR ranking function. The* ranking (condition) of a simplification rule H <=> $G \mid C \wedge B$, *where C are built-in constraints and B are CHR constraints, is the formula*

$$\forall\ (O \rightarrow rank(H) > rank(B)),$$

where O is a conjunction of order constraints implied by the built-in constraints in the rule, $\forall\ ((G \wedge C) \rightarrow O)$.

Implied order constraints such as O help to establish that rank of the lhs of the rule is strictly larger than the rank of the rhs. Since termination is undecidable for CHR, a suitable ranking and suitable order constraints cannot be found completely automatically.

3.2 Derivation Lengths

A ranking of a program satisfying the ranking condition for each rule ensures that the rank of the lhs of each rule is strictly larger than the rank of its rhs. Intuitively then, the rank of a query gives us an upper bound on the number of rule applications (derivation steps), i.e. derivation lengths. In [Fru00b] we predicted worst-case derivation lengths using rankings and compared the predictions with empirical data.

Definition 3.3 *A goal G is* bounded *if the rank of any instance of G is bounded from above by a constant.*

Theorem 3.1 *Let P be a CHR program containing only simplification rules.*

1. [Fru00a] If the ranking condition holds for each rule in P, then P is terminating for all bounded goals.

2. [Fru00b] If the ranking condition holds for each rule in P, then a worst-case derivation length D for a bounded goal G in P is the rank of G:

$$D = rank(G)$$

We are interested in rankings that get us as close as possible to the actual derivation lengths. This is the case if differences between the ranks of the lhs and rhs of the rules in a program are bounded from above. We call such rankings *tight.*

Definition 3.4 *Let S be a* ranking of a simplification rule *of the form H <=> $G \mid B$. The ranking is* exact for the rule S *iff $rank(H) = rank(B) + 1$. The ranking is* tight by n for the rule S *iff $rank(H) = rank(B) + n$. The ranking is* tight by n for a CHR program P *iff the ranking is tight by n_i for all rules in P and n is the maximum of all n_i.*

4 Worst-Case Time Complexity

We first consider the worst cost of applying a single rule, which consists of the cost to try the rule on all

constraints in the current state and of the cost to apply the rule to some constraints in the state. Then we choose the worst rule in the program and apply it in the worst possible state of the derivation. Multiplying the result with the worst-case derivation length gives us the desired upper bound on the worst-case time complexity.

In the following, we assume a naive implementation of CHR with no optimizations. The complexity of handling built-in constraints is predetermined by the built-in constraint solvers used. We assume that the time complexity of checking and adding built-in constraints is not dependent on the constraints accumulated so far in the derivation.

Lemma 4.1 *Let S be a simplification rule of the form $H <=> G \mid C \wedge B$, where H is a conjunction of n CHR constraints, G and C are built-in constraints and B are CHR constraints. The* worst-case time complexity of applying the rule *S in a state with c CHR constraints is:*

$$O(c^n(O_H + O_G) \ + \ (O_C + O_B)),$$

where O_H is the complexity of matching the lhs H of the rule, O_G the complexity of checking the guard G, O_C the complexity of adding the rhs built-in constraints C, and O_B the complexity of removing the lhs CHR constraints and of adding the rhs CHR constraints B.

Proof. *The formula consists of two summands, the first is the cost of trying the rule, the second the cost of applying the rule. In a naive implementation, we compute all possible combinations of n constraints and try to match them to the lhs of the rule. Hence, given c constraints in a query and a rule with n lhs constraints, there are $O(c^n)$ combinations of constraints to try. Each try involves matching the lhs of the rule with complexity O_H and, in the worst-case, checking the guard with complexity O_G. In the worst-case, all possible combinations have been tried before the rule is finally applied. Then, the cost of handling the rhs of the rule, $(O_C + O_B)$, is incurred.*

Now we are ready to give our meta-theorem about the time complexity of simplification rule programs. To compute the time complexity of a derivation, we have to find the worst-case for the application of a rule, i.e. the largest number of CHR constraints c_{max} of any state in a derivation and the most costly rule that could be tried and applied. We know that the number of derivation steps is bounded by the rank D. It turns out that D is also an upper bound for c_{max}.

Theorem 4.1 *Let P be a CHR program containing only constraint simplification rules. Let Q be a query with worst-case derivation length D. Then the worst-case time complexity of a derivation starting with Q is:*

$$O(D \sum_i (D^{n_i}(O_{H_i} + O_{G_i}) \ + \ (O_{C_i} + O_{B_i}))),$$

where the index i ranges over the constraint simplification rules in the program P.

Proof. *In the worst-case of a naive implementation, in each of the D derivation steps, all rules are tried on all combinations of the maximum possible number of constraints c_{max} and then the most costly rule is applied. Since rule application attempts are independent from each other, we can extend Lemma 4.1 to a set of rules in a straightforward way:*

$$O(\sum_i c_{max}^{n_i}(O_{H_i} + O_{G_i}) \ + \ Max_i(O_{C_i} + O_{B_i})),$$

where c_{max} is the worst number of CHR constraints in a derivation from a given query and Max_i takes the maximum over all i. Since the function Max and $+$ are equivalent in the O-notation, we can replace Max_i by $\sum_i$. Multiplying the resulting formula by the derivation length D yields the overall complexity:

$$O(D \sum_i (c_{max}^{n_i}(O_{H_i} + O_{G_i}) \ + \ (O_{C_i} + O_{B_i}))).$$

There cannot be more than D CHR constraints in any state of a derivation starting with a query with worst-case derivation length D, because each CHR constraint has a rank of at least 1 by definition and because each derivation step decreases the value of D by at least 1. Thus c_{max} is bounded by D. After replacing c_{max} by the bound D, we arrive at the formula of the Theorem.

From the meta-theorem it can be seen that the cost of rule tries dominates the complexity of a naive implementation of CHR.

We end this section with some general remarks on the complexities of the constituents of a simplification rule. The cost of syntactic matching O_H is determined by the syntactic size of the lhs in the given program text. Thus, its time complexity is constant.

The complexity of guard checking O_G is usually as most as high as the complexity of adding the respective constraints. The worst-case time complexity of adding built-in constraints O_C is typically linear in their size.

We assume that the complexity O_B of removing and adding CHR constraints (without applying any rules) is constant in a naive implementation where e.g. lists are used to store the CHR constraints.

5 Time Complexity of CHR Constraint Solvers

We now derive worst-case time complexities of two Boolean and one path consistency CHR constraint solver [Fru98] from the CHR library of Sicstus Prolog [HoFr98, HoFr00]. We will use *concrete syntax* of Prolog implementations of CHR, where a conjunction is a sequence of conjuncts separated by commas.

We will contrast these results with the time complexities derived from a set of test-runs with randomized data. We expect the empirical results to be better than the predicted ones, since this CHR implementation uses indexing for computing the combinations of constraints needed for lhs matching of a rule.

The Sicstus Prolog and CHR source code for the test-runs is available at `www.informatik.uni-muenchen.de/~fruehwir/chr/complexity.pl`. The code can be run via the WWW-interface of CHR Online [SaAb00].

For each solver, we will give a ranking that is an upper bound on the derivation length. From the ranking, we calculate the worst-case time complexity. We denote constant time complexity by the number 1 and zero time by 0 (this means that no computation is performed at all). We will summarize the empirical results of the test-runs in a table, see e.g. Figure 1. The tables have the following columns:

Goal the (abbreviated) goal that was run to produce the test data.

Worst the predicted worst-case derivation length D for the goal.

Apply the actual number of rule applications, i.e. derivation length.

Try the number of rules that have been tried, but not necessarily applied.

Time the time to run the goal with the CHR libarary of Sicstus Prolog, in seconds, including instrumented source code for randomization, on a recent Linux PC with medium work load.

5.1 Boolean Algebra, Propositional Logic

The domain of Boolean constraints [Me*93] includes the constants 0 for falsity, 1 for truth and the usual logical connectives of propositional logic, which are modeled here as CHR constraints. Syntactic equality = is a built-in constraint. In the constraint solver *Bool*, we simplify a single constraint $and(X, Y, X \wedge Y)$ into one or more equations whenever possible:

```
and(X,Y,Z) <=> X=0 | Z=0.
and(X,Y,Z) <=> Y=0 | Z=0.
and(X,Y,Z) <=> X=1 | Y=Z.
and(X,Y,Z) <=> Y=1 | X=Z.
and(X,Y,Z) <=> X=Y | Y=Z.
and(X,Y,Z) <=> Z=1 | X=1,Y=1.
```

For example, the first rule says that the constraint `and(X,Y,Z)`, when it is known that the first input argument `X` is 0, can be reduced to asserting that the output `Z` must be 0. Hence the goal `and(X,Y,Z),X=0` will result in `X=0, Z=0`.

Derivation Length. Since a single rule application reduces each CHR constraint to built-in constraints, the worst-case derivation length is just the number of constraints in the query, c. Let the ranking be

$$rank(A) = 1 \text{ if } A \text{ is an atomic CHR constraint}$$

For each rule in *Bool*, H <=> G | B, we have that $rank(H) = 1$ and $rank(B) = 0$. Hence the ranking is exact for all rules. Consequently, the worst-case derivation length of a Boolean goal is

$$D_{Bool} = c$$

It can be much smaller. For example, the goal `and(U,V,W)` delays, its derivation length is zero. Another example is a goal that contains the constraint `and(0,Y,1)`. If it is selected first, it will reduce to the inconsistent built-in constraint `1=0` in one derivation step. Because of the inconsistency, this is a final state of the derivation.

Complexity. All rules have one lhs CHR constraint, i.e.. $n = 1$. The derivation length is bounded by c. Checking or establishing built-in syntactic equality between variables and the constants 0 and 1 can be implemented in constant time. Then, for all rules, (O_H, O_G, O_C, O_B) is $(1, 1, 1, 0)$, i.e. all rule-dependent complexities are constant. According to the complexity meta-theorem, this gives $O_{Bool}(c(c^1(1 + 1) + (1 + 0)))$, i.e

$$O_{Bool}(c^2)$$

Empirical Results. In Figure 1, the Prolog predicate `tst/3` produces a chain of and constraints, where the last variable of one constraint is the first variable of the next constraint. The first (`A`) and the last (`B`) variable are returned in the second and third argument of `tst/3`, respectively.

Figure 1 shows that

- The order of (built-in) constraints may strongly influence the run-time.

Goal	Worst	Apply	Try	Time
tst(125,A,B), A=1	125	1	759	0.06
tst(250,A,B), A=1	250	1	1509	0.10
tst(500,A,B), A=1	500	1	3009	0.22
tst(1000,A,B), A=1	1000	1	6009	0.43
tst(2000,A,B), A=1	2000	1	12009	0.87
tst(4000,A,B), A=1	4000	1	24009	1.73
tst(8000,A,B), A=1	8000	1	48009	3.46
tst(125,A,B), B=1	125	125	1500	0.11
tst(250,A,B), B=1	250	250	3000	0.24
tst(500,A,B), B=1	500	500	6000	0.47
tst(1000,A,B), B=1	1000	1000	12000	0.95
tst(2000,A,B), B=1	2000	2000	24000	1.88
tst(4000,A,B), B=1	4000	4000	48000	3.75
tst(8000,A,B), B=1	8000	8000	96000	7.51
tst(125,A,B), A=0	125	125	875	0.07
tst(250,A,B), A=0	250	250	1750	0.15
tst(500,A,B), A=0	500	500	3500	0.29
tst(1000,A,B), A=0	1000	1000	7000	0.57
tst(2000,A,B), A=0	2000	2000	14000	1.16
tst(4000,A,B), A=0	4000	4000	28000	2.34
tst(8000,A,B), A=0	8000	8000	56000	4.67
A=0, tst(125,A,B)	125	125	125	0.01
A=0, tst(250,A,B)	250	250	250	0.02
A=0, tst(500,A,B)	500	500	500	0.03
A=0, tst(1000,A,B)	1000	1000	1000	0.06
A=0, tst(2000,A,B)	2000	2000	2000	0.11
A=0, tst(4000,A,B)	4000	4000	4000	0.21
A=0, tst(8000,A,B)	8000	8000	8000	0.44

Figure 1: Results from Test-Runs with Boolean And

- The actual derivation length reaches the predicted worst-case derivation length.
- The number of rule applications may be arbitrarily small.
- The number of rule tries is up to 12 times larger than the worst-case derivation length. Note that there are 6 rules.
- Run-time is linear in the number of rule tries.

These observations will also hold for the other constraint solvers we considered, except of course for the relationship between the number of rule applications and the number of rule tries.

In practice, the observed time complexity of the solver seems to be linear:

$$O^{obs}_{Bool}(c)$$

We attribute this difference to the effect of indexing on variables which allows to find matching constraints faster.

Boolean Cardinality

The cardinality constraint combinator was introduced in the CLP language cc(FD) [vHSD95] for finite domains. In the solver *Card* we adapted cardinality for Boolean variables. The Boolean cardinality constraint `#(L,U,BL,N)` is true if the number of Boolean variables in the list `BL` that are equal to 1 is between `L` and `U`. `N` is the length of the list `BL`. Boolean cardinality can express negation `#(0,0,[C],1)`, exclusive or `#(1,1,[C1,C2],2)`, conjunction `#(N,N,[C1,...,Cn],N)` and disjunction `#(1,N,[C1,...,Cn],N)`.

```
% trivial, positive, negative satisfaction
triv_sat@ #(L,U,BL,N) <=> L=<0,N=<U | true.
pos_sat @ #(L,U,BL,N) <=> L=N | all(1,BL).
neg_sat @ #(L,U,BL,N) <=> U=0 | all(0,BL).

% positive and negative reduction
pos_red @ #(L,U,BL,N) <=> delete(1,BL,BL1) |
                          0<U, #(L-1,U-1,BL1,N-1).
neg_red @ #(L,U,BL,N) <=> delete(0,BL,BL1) |
                          L<N, #(L,U,BL1,N-1).
```

In this CHR program, all constraints except cardinality are built-in. `all(B,L)` equates all elements of the list `L` to `B`. `delete(X,L,L1)` deletes the element `X` from the list `L` resulting in the list `L1`. Due to the semantics of guard checking, `X` must exactly match the element that is to be removed.

Derivation Length. Our ranking is based on the length of the list argument of the Boolean cardinality constraint:

$$rank(\#(L,U,BL,N)) = 1 + length(BL)$$
$$length([]) = 0$$
$$length([X|L]) = 1 + length(L)$$

$$delete(X,L,L1) \rightarrow length(L) = length(L1)+1$$

The rank adds one to the length of the list in order to give a cardinality with the empty list a positive rank. For example, the goal `#(0,0,[],0)` has derivation length one (no matter which of the three satisfaction rules is applied). Let the rank of atomic constraints in a query be bounded by a constant l.

The ranking is exact for the two recursive reduction rules, because of the order constraint implied by `delete`. It is tight by l only for the three satisfaction rules, since a cardinality constraint with arbitrary rank may be reduced to built-in constraints with rank 0 in one derivation step. Hence the ranking of the solver program *Card* is tight by l.

From the ranking we see that the derivation length of a single cardinality constraint is bounded by the length of the list argument. For example, the goal `#(1,1,[0,0,0,0,X],5)` needs five derivation steps to reduce to `X=1`. The first four steps remove the zeros from the list. The derivation length of a goal is less or equal to the sum of the lengths of the lists occurring in the goal. Hence it is linear in the syntactic size of the goal in the worst-case.

Let c be the number of CHR constraints in a query and recall that the rank of atomic constraints is bounded by l. Then we have that:

$$D_{Card} = cl$$

Complexity. All rules have one lhs CHR constraint. Time complexity for the built-in constraints `delete` and `all` can be assumed to be linear in the length of the list, and is constant for the other built-in constraints. The derivation length is bounded by cl. The time complexities, (O_H, O_G, O_C, O_B), of the five rules are $(1,1,1,0), (1,1,l,0), (1,1,l,0), (1,l,1,1)$ and $(1,l,1,1)$ respectively. Hence the complexity for both the rule tries and the rule applications is at worst linear in l. According to the meta-theorem, the time complexity is $O(cl((cl)^1 l + l))$, i.e.

$$O(c^2 l^3).$$

Empirical Results. Our empirical results are presented in Figure 2. *allr* is a variation on *all*, it starts equating the list elements from the back of the list. This means that in the guards of the recursive rules for cardinality, `delete` has to search till the end of the list to find a zero or one. *card_rand* produces a random list of variables, zeros and ones, each of the three with the same probability. The list lengths were chosen at random between 0 and 1000 and then the problem instances were ordered by list length.

The table shows that

- For the `card_rand` problem, the figures follow from to the probability distribution.
- The other problem instances show the influence of the order of built-in constraints on the run-time. However, timings differ by a constant factor, so complexity is not affected.
- The number of rule tries is up to 10 times larger than the worst-case derivation length. Note that there are 5 rules, and they may be tried in vain.
- Run-time is roughly quadratic in the list length.

Goal	Worst	Apply	Try	Time
card_rand(N,A,B,L), #(A,B,L,N)	40	30	132	0.03
	91	56	260	0.12
	217	143	655	0.71
	298	199	901	1.25
	655	450	2029	5.07
	672	446	2008	5.10
#(0,1,L,N),all(0,L)	109	108	1071	0.65
	200	199	1981	1.98
	318	317	3161	4.01
	382	381	3801	5.27
#(0,1,L,N),allr(0,L)	109	108	1071	0.69
	200	199	1981	2.38
	318	317	3161	4.47
	382	381	3801	7.72
#(0,1,L,N),all(X,L), X=0	109	108	1076	0.34
	200	199	1986	0.99
	318	317	3166	2.29
	382	381	3806	3.21
all(0,L),#(0,1,L,N)	109	108	536	0.14
	200	199	991	0.45
	318	317	1581	1.12
	382	381	1901	1.58

Figure 2: Test-Run Results for Boolean Cardinality

We also did some experiments with more than one cardinality constraint but found that the overall run-time was the sum of the run-times of each constraint alone. Thus the observed time complexity has lower exponents than those predicted:

$$O^{obs}_{Card}(cl^2)$$

We again attribute this difference to the effect of indexing on variables.

5.2 Path Consistency

In this section we analyze a constraint solver that implements the classical artificial intelligence algorithm of path consistency [MaFr85, MoHe86].

A *disjunctive binary constraint* $c(I, J, \{r_1, \ldots, r_n\})$, also written $I\ \{r_1, \ldots, r_n\}\ J$, is a finite disjunction $(I\ r_1\ J) \vee \ldots \vee (I\ r_n\ J)$, where each r_i is a binary relation. The r_i are called *primitive constraints*. The number p of primitive constraints is finite and they are pairwise disjoint.

W.l.o.g. we will assume that in a query, for each ordered pair of variables, there is a disjunctive binary constraint between them. The basic operation of path consistency computes a tighter constraint between two variables I and J by intersecting it with the constraint composed from the two constraints between I and a third variable K and between K and J. This operation can be implemented directly by a single rule in the solver *Path*:

```
path_consistency @
c(I,K,C1), c(K,J,C2), c(I,J,C3) <=>
    composition(C1,C2,C12),
    intersection(C12,C3,C123),
    C123=\=C3 |
    c(I,K,C1), c(K,J,C2), c(I,J,C123).
```

The repeated application of the rule will make the initial query constraints path consistent. The built-in constraints `composition` and `intersection` implement functions on pairs of disjunctive binary constraints:

composition(C_1, C_2, C_3) iff
$I\ C_1\ K\ \wedge\ K\ C_2\ J \rightarrow I\ C_3\ J$,
where C_3 is the smallest set of primitive constraints implied for given C_1 and C_2.

intersection(C_1, C_2, C_3) iff
$I\ C_1\ J\ \wedge\ I\ C_2\ J \leftrightarrow I\ C_3\ J$.

The check `C123=\=C3` makes sure that the new constraint `C123` is different from the old one `C3`.

Derivation Length. We rely on the following ranking:

$$rank(c(I,J,C)) = 1 + card(C)$$
$$card(\{c_1,\ldots,c_n\}) = n$$

$$intersection(C1,C2,C3) \rightarrow card(C3) \leq card(C1) \wedge card(C3) \leq card(C2)$$
$$intersection(C1,C2,C3) \wedge C3 \neq C2 \rightarrow card(C3) \neq card(C2)$$

For the ranking, one is added to the cardinality of C so that constraints with an empty set C have a positive rank as well. Queries are bounded, when C is known.

Because of the properties of intersection and the guard check `C123=\=C3`, the cardinality of `C123` must be strictly less than that of `C3`. Hence the rhs is ranked strictly smaller than the lhs of the rule. Every rule application removes at least one primitive constraint and at most all of them from the set of primitive constraints `C3` by intersecting it with `C12`. Hence, if the maximum number of primitive constraints is p, the ranking is tight by p. The worst-case derivation length is linear in the syntactic size of the goal:

$$D_{Path} = cp$$

Complexity. There is one rule, it has three lhs CHR constraints. For small p, the built-in constraints for composition, intersection and inequality checking can be implemented by table look-up, i.e. in constant time. Otherwise, we define the operations in terms of primtive constraints. Composition of disjunctive constraints can be computed by pairwise composition of its primitive constraints. Intersection for disjunctive constraints can be implemented by set intersection, since primitive constraints are disjoint. We assume constant time access to individual elements in the composition table of primitive constraints. Then composition can be implemented in quadratic time, $O(p^2)$. Intersection and inequality checking can be implemented in linear time.

Hence, according to the meta-theorem, the complexity is $O(cp((cp)^3(1+(p^2+p+p))+(0+1)))$, i.e

$$O_{Path}(c^4p^6)$$

Empirical Results. In the goals of Figure 3, `tpath` generates constraints between each pair of different variables in its argument list. The disjunctive constraints C are randomly chosen non-empty subsets of $\{<,=,>\}$, each with the same probability. Hence p is a constant, $p = 3$. For a list of length n, there are exactly $c = n(n-1)$ constraints. Thus the worst case derivation length is $3n(n-1)$. The table entries have been sorted.

Goal	Worst	Apply	Try	Time
length(L,V),	168	32	1079	0.44
tpath(L,A),	168	41	1151	0.50
V=8	168	45	1477	0.65
	168	49	1279	0.55
..., V=12	396	87	4024	1.76
	396	101	4791	2.11
	396	102	4622	2.00
	396	104	4895	2.12
..., V=16	720	155	10241	4.54
	720	155	10724	4.82
	720	160	11709	5.23
	720	185	12330	5.54
..., V=20	1140	241	22075	10.33
	1140	263	23578	11.04
	1140	269	24154	11.24
	1140	277	23573	11.02

Figure 3: Test-Run Results for Path Consistency

The table shows that

- The actual derivation length is roughly linear in the predicted worst-case derivation length, i.e. linear in the number of constraints.
- The number of rule tries increases faster than the worst-case derivation length.

- Run-time is roughly linear in the number of rule tries. It is roughly cubic in the number of variables v.

We can conclude from the current experiments, where p is constant, that the observed complexity is much lower than the predicted one. Since $O(v^2) = O(c)$ we have:

$$O^{obs}_{Path}(v^3) = O^{obs}_{Path}(c^{1.5})$$

For $p = 3$, this corresponds to the complexity of the best known general algorithm for path consistency, which is $O(v^3p^3)$ [MaFr85, MoHe86].

6 Conclusions

Based on the worst-case derivation length, as given by a ranking, we were able to give a general complexity meta-theorem for the worst-case time complexity of CHR constraint simplification rule programs. Rankings were originally used to prove termination. They map constraints and terms to natural numbers such that the rank of the lhs of a rule is larger than the rank of the rhs of a rule. Once a ranking has been found, our meta-theorem allows for computing the complexity automatically from the program text. Our theorem also applies to naive implementations of CHR simplification rules.

We have found that the dominating factor in the complexity are the rule application attempts (rule tries), not the actual rule applications. The cost of rule tries depends on the number of lhs CHR constraints n, the complexity of the guard checking and the ranking D of a given query. D was bounded by the product cr, where c is the number of atomic CHR constraints in the query and r is the maximum rank of an atomic CHR constraint in the query. r often can be interpreted as syntactic size. Built-in constraints only contribute if they have non-constant complexity. This is the case if non-scalar datatypes like lists or sets are involved. In our examples, the derived complexities were of the form $c^{n+1}r^{n+1+k}$, where k is a small constant (often zero) introduced by the built-in constraints.

We compared the complexities predicted by our theorem with the complexities observed in empirical tests of two Boolean and a path consistency constraint solver. Due to optimizations like indexing on variables in the Sicstus Prolog CHR implementation, the observed complexities were better than the predicted ones.

They involved the same parameters, but lower exponents. In the case of the two Boolean constraint solvers, the complexity of rule tries was lowered to the complexity of rule applications. In an instance of the path consistency solver we observed a complexity that corresponds to the complexity of the best known algorithm for the problem. This solver consists of just one rule. Although we tried to produce examples that would exhibit the worst case behavior of the implementation, the empirical results are preliminary. At this stage of the research we cannot rule out with certainty that there are cases were the implementation actually shows the predicted worst-case complexity. Clearly more experiments are necessary.

Further work should take into account the effect of indexing and other optimizations in the complexity predictions. Another open question is which aspects in finding an appropriate ranking can be automated. We also would like to extend our approach to propagation rules. The difficulty is that for propagation rules, the ranking approach for derivation lengths does not apply. The approach of [McA99, GaMc01] also does not apply, since it does not deal with free variables at run-time and arbitrary built-in constraints.

Acknowledgements

Part of this work was performed while visiting the School of Computer Science and Software Engineering at Monash University, Melbourne, in March 2000 and while visiting the Department of Computer Science at the University of Pisa in October 2001.

References

[AAI00] Applied Artificial Intelligence, Special Issue on Constraint Handling Rules (C. Holzbaur and T. Frühwirth, Eds.), Taylor & Francis, Vol 14(4), 2000.

[AbFr99] S. Abdennadher and T. Frühwirth, Operational Equivalence of CHR Programs and Constraints, 5th Intl Conf on Principles and Practice of Constraint Programming (CP'99), Springer LNCS 1894, 1999.

[AFM99] S. Abdennadher, T. Frühwirth and H. Meuss, Confluence and Semantics of Constraint Simplification Rules, Constraints Journal Vol 4(2), Kluwer Academic Publishers, 1999.

[Abd97] S. Abdennadher, Operational Semantics and Confluence of Constraint Propagation Rules, 3rd Intl Conf on Principles and Practice of Constraint Programming (CP'97), Springer LNCS 1330, 1997.

[Abd00] S. Abdennadher, A Language for Experimenting with Declarative Paradigms, Second Workshop on Rule-Based Constraint Reasoning and Programming, at the 6th Intl Conf on Principles and

Practice of Constraint Programming (CP'2000) Singapore, 2000.

[Fru98] T. Frühwirth, Theory and Practice of Constraint Handling Rules, Special Issue on Constraint Logic Programming (P. J. Stuckey and K. Marriot, Eds.), Journal of Logic Programming Vol 37(1-3), Elsevier, 1998.

[Fru00a] T. Frühwirth, Proving Termination of Constraint Solver Programs, in New Trends in Constraints, (K.R. Apt, A.C. Kakas, E. Monfroy and F. Rossi, Eds.), Springer LNAI 1865, 2000.

[Fru00b] T. Frühwirth, On the Number of Rule Applications in Constraint Programs, Declarative Programming - Selected Papers from AGP 2000, (A. Dovier, M. C. Meo, A. Omicini, Eds.), Electronic Notes in Theoretical Computer Science (ENTCS), Vol 48, Elsevier Science Publishers, June 2001.

[Fru01] T. Frühwirth, As Time Goes by: Complexity Analysis of Simplification Rules, Workshop on Quantitative Aspects of Programming Laguages (QAPL'01), at the Conf on Principles, Logics, and Implementations of high-level programming languages (PLI'01), Firenze, Italy, September 2001.

[GaMc01] H. Ganzinger and D. McAllester, A New Meta-Complexity Theorem for Bottum-up Logic Programs, (R. Gore, A. Leitsch and T. Nipkow, Eds.) First Intl Joint Conf on Automated Reasoning IJCAR 2001, Springer LNAI 2083, 2001.

[HoFr98] C. Holzbaur and T. Frühwirth, Constraint Handling Rules Reference Manual for Sicstus Prolog, TR-98-01, Österreichisches Forschungsinstitut für Artificial Intelligence, Vienna, Austria, July 1998.

[HoFr00] C. Holzbaur and T. Frühwirth, A Prolog Constraint Handling Rules Compiler and Runtime System, Applied Artificial Intelligence, Special Issue on Constraint Handling Rules (C. Holzbaur and T. Frühwirth, Eds.), Taylor & Francis, Vol 14(4), 2000.

[JaMa94] J. Jaffar and M. J. Maher, Constraint Logic Programming: A Survey, Journal of Logic Programming Vol 19,20, Elsevier, 1994.

[MaFr85] A. K. Mackworth and E. C. Freuder, The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems, Artificial Intelligence Vol 25, 1985.

[MaSt98] K. Marriott and P. J. Stuckey, Programming with Constraints, MIT Press, USA, 1998.

[McA99] D. McAllester, On the Complexity Analysis of Static Analyses, (A. Cortesi and G. File, Eds.), 6th Intl. Static Analysis Symposium (SAS'99), Springer LNCS 1694, 1999.

[Me*93] S. Menju et al., A Study on Boolean Constraint Solvers, Constraint Logic Programming: Selected Research, (F. Benhamou and A. Colmerauer, Eds.), MIT Press, Cambridge, Mass., USA, 1993.

[MoHe86] R. Mohr and T.C. Henderson, Arc and Path Consistency Revisited, Artificial Intelligence 28, 1986.

[SaAb00] M. Saft and S. Abdennadher, Web-CHR, Ludwig-Maximilians-Universität München, `www.pms.informatik.uni-muenchen.de/~webchr`, March 2000.

[vHSD95] P. van Hentenryck, V. A. Saraswat, and Y. Deville, Constraint Processing in cc(FD), Chapter in Constraint Programming: Basics and Trends, (A. Podelski, Ed.), Springer LNCS 910, 1995.

Inference complexity as a model-selection criterion for learning Bayesian networks

Alina Beygelzimer
Department of Computer Science
University of Rochester
Rochester, NY 14627
beygel@cs.rochester.edu

Irina Rish
IBM T. J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532
rish@us.ibm.com

Abstract

Learning Bayesian networks from data is commonly viewed as a (constrained) optimization problem, where a particular scoring metric is used for evaluating the quality of a candidate model. Commonly used scoring metrics, such as BIC/MDL, prefer models that fit the data well and have low representation complexity (i.e. the number of parameters). In this paper, we argue that a model selection criteria must also include another important property of Bayesian networks, namely their *inference complexity*, which is exponential in *treewidth* of the underlying graph. We demonstrate that traditional metrics may not distinguish between networks with drastically different treewidths. In particular, we show that it is quite likely to have networks that represent close distributions (in terms of KL-divergence) and have similar representation complexity, but fairly different treewidths. We also investigate the relationships between the treewidth and other structural properties of graphs that suggest efficient probabilistic tests for a quick estimation of the treewidth during learning.

Keywords: *Bayesian networks, learning, efficient inference, treewidth.*

1 Introduction

This paper is motivated by the goal of learning Bayesian networks that allow *efficient inference*, as opposed to learning the networks first without consideration of their inference complexity and *then* dealing with this complexity by means of approximation. As we demonstrate in this paper, traditional model selection criteria that aim at fitting the data and minimizing the *representation* complexity are unable to capture the *inference* complexity, and thus must be extended appropriately.

A Bayesian network (BN) is a graphical probabilistic model that exploits conditional independencies among a set of random variables $\mathbf{X} = \{X_1, ..., X_n\}$ to provide a compact product-form representation of their joint probability distribution $P(\mathbf{X})$. The *structure* of a BN is a directed acyclic graph (dag), denoted $G = (V, E)$, where the nodes in V correspond (by one-to-one mapping) to the variables[1] and the edges encode probabilistic dependencies. Every node X_i is associated with a local conditional probability distribution (CPD) $\mathbf{P}(X_i \mid \Gamma(X_i))$, where $\Gamma(X_i)$ denotes the set of *parents* of node X_i (i.e., nodes pointing to X_i) in the graph; a node without parents is associated with its prior distribution $\mathbf{P}(X_i)$. Thus, a Bayesian network is a pair (G, Θ) where G is its dag, and Θ is the set of all CPD parameters. The joint distribution encoded by the network is given by the product of the CPDs: $\mathbf{P}(X_1, ..., X_n) = \prod_{i=1}^{n} \mathbf{P}(X_i \mid \Gamma(X_i))$. By the *representation complexity* of a particular Bayesian network structure we mean the number of *independent parameters* (i.e., the minimal number of conditional and prior probabilities) needed to specify the network. For example, to specify the CPD for a node X with k parents we need 2^k probabilities $P(X = 0 \mid \gamma(X))$, where $\gamma(X)$ is a particular assignment to the parents of X. Given a network structure G, let $\mu(G)$ denote the representation complexity of G.

Bayesian networks are used for *probabilistic inference*, i.e. for answering probabilistic queries, such as *belief updating*, or finding the posterior probabil-

[1] A common assumption in Bayesian network literature is that random variables have a finite set of values. In this paper, we particularly focus on networks with binary variables; extending the results to general finite domains is straightforward.

ity $P(\mathbf{Y}|\mathbf{Z}=\mathbf{z})$ of an unknown set of variables $Y \subseteq \mathbf{X}$ given *evidence* $\mathbf{Z}=\mathbf{z}$, or finding the most-likely variable assignment $\mathbf{y}' = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{Z}=\mathbf{z})$, called the *maximum aposteriory probability (MAP)* assignment. However, (exact) probabilistic inference in Bayesian networks is NP-hard [4]. Commonly used inference algorithms, such as the *junction tree* (or *join-tree*) algorithm [15] and closely related to it *variable-elimination* techniques [5, 26, 6], are exponential in the size (the dimensionality) of largest dependence recorded by the algorithm, which is equivalent to the size of largest clique (minus 1) "induced" in the *moral graph* of BN[2]. This graph parameter is called the *induced width*, or the *treewidth* [7], and is formally defined as follows. Given a (directed or undirected) graph G, the *width* of X_i along ordering o is the number of X_i's neighbors preceding X_i in o. The *width of the graph* along o, denoted w_o, is the maximum width along o. The *induced graph* of G along o is obtained by connecting the preceding neighbors of each X_i, going from $i=n$ to $i=1$. The induced width along o, denoted w_o^*, is the width of the induced graph along o, while the induced width w^* is the minimum induced width along any ordering.

Learning Bayesian networks from data has recently become an active area of research [11]. A typical approach is based on defining (1) a scoring metric for evaluating the quality of a given structure, and (2) a search strategy for traversing the space of candidate models using some local transformations (e.g. adding, deleting, or reversing an edge). A learning algorithm then searches for a model that optimizes the scoring metric. Related approaches may also use some prior knowledge about the domain (e.g. marginal independence between certain variables) to constrain the search.

Commonly used information-theoretic scoring metrics such as BIC or MDL, AIC, MML and others, provide a tradeoff between the accuracy and the complexity of a model by minimizing the sum of the encoding length of the model (i.e. its representation complexity) and the encoding length of the data given the model, which is measured by the negative log-likelihood and thus reflects the model error. Formally, the *minimum description length* (MDL) [22] score of a Bayesian network $B=(G,\Theta)$ (which is equivalent to Bayesian information criterion, or BIC), is defined as $\mathrm{MDL}(G,\Theta \mid D) = \frac{\log N}{2}\mu(G) - \log \mathbf{P}(D \mid G,\Theta)$, where D is a set of N observations (training data), and $\mu(G)$ is the number of independent parameters needed to specify the CPDs. The first term of the MDL score is the description length of the model, while the second term, the negative log-likelihood of the model given data, is the number of bits needed to describe the data given the model. Thus MDL favors models that predict data better (have higher log-likelihood) and have lower representation complexity. However, it is independent of other important properties of the learned model, in particular, its inference complexity. For example, we may learn two networks that fit the data equally well, and have similar representation complexity, yet their treewidth (and therefore, their inference complexity) is quite different. In this paper, we provide asymptotic (with increasing networks size) theoretical analysis and some empirical evidence suggesting that such situations are frequent enough to argue for using the treewidth test during learning. We demonstrate that two networks having drastically different treewidths may quite often represent close distributions (in terms of KL-divergence) and have similar number of parameters (and thus similar BIC/MDL scores). Moreover, we describe a construction of two large natural classes of BNs that have this property.

In this paper, we argue that learning algorithms should incorporate treewidth as a part of their model selection criteria. Clearly, testing the treewidth of a candidate model during learning must be very efficient. Computing the exact treewidth of a given graph is known to be NP-hard, although there are polynomial-time heuristics for upper bounding the treewidth [7, 1]. However, we do not need to compute the treewidth exactly, nor we need a good approximation of it; it suffices to have a quick probabilistic test that asserts whether the treewidth is "small enough" or very large (i.e. far from being "small enough"). We want the complexity of such test to be independent of n, depending only on the quality of this assertion and the parameter quantifying "small enough". Furthermore, since the search is usually based on local transformations of the structure (e.g. adding, deleting, or reversing a single edge), the treewidth test can be incremental. We initiate an investigation of the relationships between structural properties of graphs and the complexity of inference in the graphs. In particular, we explore the relationships between the number of parameters needed to specify the network and the treewidth of the network. These investigations suggest efficient probabilistic tests for predicting the treewidth from other (easy-to-compute) graph invariants without actually computing it directly.

[2]The *moral* graph of G results from connecting ("marrying") all the parents of every node in G by an undirected clique, then dropping the directionality of other edges. Moral graphs basically allow to represent the set of dependencies encoded by the original directed graph without the use of edge directionality. (Notice that every pair of nodes participating in the same CPD is connected in the moral graph).

2 Treewidth difference of similar distributions

In this section, we outline our preliminary results suggesting that it is quite common to encounter two Bayesian network models that represent similar distributions and yet have very different treewidths[3]. We show that there are many networks that encode close distributions (measured by their *KL-divergence*), but have a relatively large difference in the treewidth. The *KL-divergence* between two probability distributions p and q is defined as $KL(p,q) = \mathrm{E}[\log p/q]$, where the expectation is taken with respect to p. We will use the "symmetrical" KL-divergence defined as $d(p,q) = KL(p,q) + KL(q,p)$.

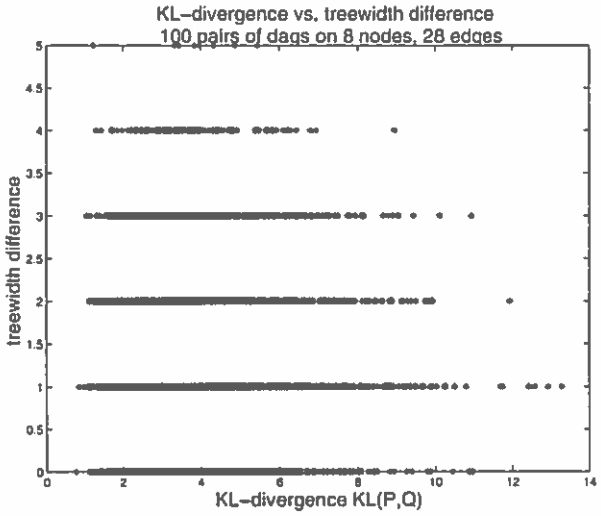
Figure 1(a) shows KL-divergence versus treewidth difference for 100 pairs of randomly generated Bayesian networks on $n = 8$ nodes with $\binom{n}{2} = 28$ edges. We observe no correlation between the treewidth difference and the KL-divergence: close distribution may have quite different treewidth, and vice versa.

Furthemore, we provide a way of constructing BNs that represent close distributions whose treewidths can be very different. Given an "initial" BN $B = (G, \Theta)$ representing a joint probability distribution $P(\mathbf{X})$, we will construct a family of BNs by significantly modifying G, and at the same time keeping a small distance between $P(\mathbf{X})$ and its projection (in terms of symmetric KL-divergence) onto the set of distributions consistent with a modified G. The main idea here is that removal (or addition) of "weak" edges, i.e. the edges corresponding to "weak" probabilistic dependencies will not change much the distribution but may significantly change the underlying graph G. The strength of an edge between the variable X_i and its parent Y, denoted as $I_i(Y)$, can be quantified, for example, by the mutual information between the two variables, as proposed in [18] (see [3] for details).

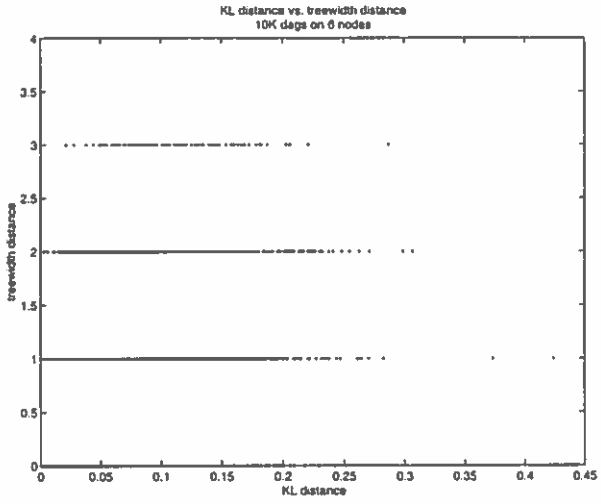
Definition 1 *Given a Bayesian network $B = (G, \Theta)$ containing a node X_i and its parent Y, an edge $Y \to X_i$ is said to be* δ-strong *(for $0 < \delta < 1$) if $I_i(Y) \geq \delta$, where $I_i(Y)$ is a measure of Y's influence on X_i. The* δ-core *of a B, denoted G_δ, is the subgraph of G containing all δ-strong edges and all vertices of G.*

The cores suggest a way of constructing BNs with the desired property. Figure 1(b) shows empirical

[3]Note that our analysis is asymptotic, i.e. we assume sufficiently large data sets, so that we can compare the distributions represented by competing models directly, instead of comparing the corresponding data likelihoods used in model-selection criteria.



(a)



(b)

Figure 1: (a) Treewidth difference (y-axis) versus KL-divergence (x-axis) for 100 pairs of randomly generated Bayesian networks (8 nodes, 28 edges). (b) Treewidth difference (y-axis) versus symmetric KL-divergence (x-axis) on 10,000 Bayesian networks with 6 nodes constructed using the "strong-core" technique.

results from generating networks with strong cores (containing most of the probability mass of the corresponding distributions) and weak symmetric differences. This allows to keep the distributions close and, at the same time, to have a large difference in the treewidth. Clearly, the properties of the network we starts with, determine the limits of the family this network spawns. For example, if the distribution of arc weights (defined in terms of mutual information [18]) is highly non-uniform (i.e. only a few terms make large contributions to the multi-information encoded by the network), then the network must have many "weak" edges that can be removed without losing much accuracy. In our construction (see Figure 1(b)), we started with random dags; this was good enough to give networks with small KL-distance, yet largely different graph structures. Of course, our analysis can only be considered as preliminary and requiring further experiments on larger graphs; we believe that with increasing graph size, we will often observe the increasing gap in the treewidth among BNs representing similar distributions.

3 Representation complexity versus inference complexity

In this section, we focus on the representation complexity part of the model selection criteria. We demonstrate its inability to capture the inference complexity by constructing two parametric families of Bayesian networks that have (asymptotically) similar representation complexity (the number of parameters), but drastically different inference complexity (treewidth).

Notice that a small representation complexity does not imply a small treewidth. Consider a simple, well-known example of the $n \times n$ *grid*, where the direction of all vertical (horizontal) edges is from left to right (from top to bottom respectively). If the nodes are binary, the grid requires less than $4n^2$ parameters. However, it is easy to see that the treewidth of its moral graph is n (notice that the moral graph is already triangulated); hence the inference complexity is exponential in n. The grid was used in [2] to prove the following proposition.

Proposition 1 [2] *For every $n > 3$, there exist dags $G = G(n)$ and $G' = G'(n)$ such that $\mu(G) > \mu(G')$, whereas $w^*(G) = 2$, while $w^*(G') = \sqrt{n}$.*

Proof: See Appendix. ∎

Now we give a construction promised above. In fact, it will be sufficient to restrict ourselves to bipartite dags, an important family of structures that often arise in various diagnosis and prediction problems. For example, one can think of a bipartite graph as a diagnosis network with one component representing independent unknown variables to be diagnosed (e.g., diseases or computer faults) and the other component representing the observed symptoms caused by those unknown variables (see, for example, the QMR-DT network for medical diagnosis [23], or the BN for computer network fault diagnosis [21]).

3.1 Construction

We need to exhibit two families of dags with the same parameter complexity: an *easy*-family $\mathcal{G}' = \{\mathcal{G}'_n\}$ with small treewidth, and a *hard*-family $\mathcal{G}'' = \{\mathcal{G}''_n\}$ with large treewidth, where $\mathcal{G}'_n$ and $\mathcal{G}''_n$ are the families restricted to n-node dags. Both families will be non-explicit (i.e. given by a probabilistic construction). In fact, both families will be instances of the same parameterized construction, which we define next.

Let the (l, r, d)-*diffuser* be a directed bipartite graph D with l "left" nodes, each of degree d, and r "right" nodes, such as the direction of all edges is from right to left. Define a *random* (l, r, d)-diffuser, constructed by connecting every left node to a randomly chosen subset of d right nodes [4].

Consider such a random diffuser D with parameters $(l = n - 2k, r = 2k, d = k)$. Let $\mathcal{G}'_n$ be the probability space of D, i.e. the space of all bipartite graphs constructed as above. Clearly, the treewidth of the moralized D is at most $2k - 1$. We also have $\mu(D) = (n - 2k)2^k + 2k$. The number of graphs in the family is $\binom{2k}{k}^{n-2k} \sim (\frac{2^{2k}}{\sqrt{2k}})^{n-2k}$. For $k = \log n$, we have $|\mathcal{G}'_n| = O(n^{n/2})$, $\mu = O(n^2)$, and $w^* = O(\log n)$.

Now let the hard family $\mathcal{G}''_n$ be the probability space of a random $(l = n - n^{2/3}, r = n^{2/3}, d = \log n)$-diffuser. We have $|\mathcal{G}''_n| = r^{dl} = O(n^{n \log n})$, which gives a large family of n-node dags with at most $\mu = n^{2/3} + n(n - n^{2/3}) = O(n^2)$ parameters. We will show that the treewidth of the moral graph of such a diffuser is at least $n^{2/3} - 1$.

Lemma 1 *With probability at least $1 - exp(-\frac{ld^2}{r^2} + 2 \ln r)$, a random (l, r, d)-diffuser has treewidth $r - 1$.*

[4] Random diffusers are equivalent to the graph structure of *low-density parity-check codes (Gallager codes)* [17], currently considered to be the state-of-the-art codes when combined with the appropriate probabilistic decoder [10]. Gallager codes transform a block of r input bits into l additional parity-check bits, each parity-check bit is computed for a subset of d randomly selected input bits.

Proof: See Appendix. ∎

In particular, for $l = n - n^{2/3}$, $r = n^{2/3}$, and $d = \log n$, we have $w^* \geq n^{2/3} - 1$ with probability at least $1 - exp(-n^{1/2}\log^2 n)$. We have defined two large families of bipartite dags such that the graphs in both families have $O(n^2)$ parameters, yet the graphs in one family have treewidth $O(\log n)$ while the graphs in the other family have treewidth $O(n^{2/3})$. Notice that the treewidth ratio in this case is $O(n^{2/3 - \frac{\log\log n}{\log n}})$, which is asymptotically larger than the ratio $O(n^{1/2})$ obtained in Proposition 1.

3.2 Bounds on treewidth

A natural question that arises is that of the exact relation between $\mu(\cdot)$ and $w^*(\cdot)$. In this section we examine the distribution of n-node dags according to their number of parameters. Clearly, the maximum complexity is that of the clique (when all variables are pairwise dependent), in which case $\mu = 2^n - 1$.

Figure 2 shows how $w^*(\cdot)$ is distributed as a function of $\mu(\cdot)$, i.e. it shows all possible combinations of (μ, w^*) pairs over all non-isomorphic dags on a fixed number of nodes.

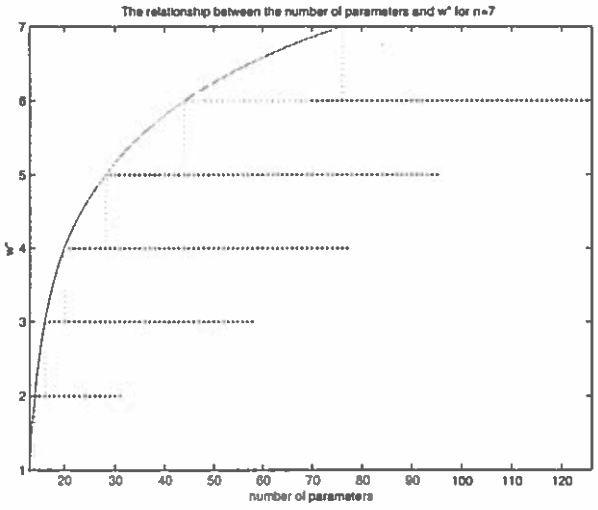


Figure 2: Relationship between $\mu(\cdot)$ and $w^*(\cdot)$. The solid curve corresponds to the upper bound; the corresponding integer upper bound is shown as a step-wise dotted line).

We derive bounds on w^*, which explain the bounding curves in Figure 3.2. Moreover, we show that both the lower and the upper bound below may be attained.

Proposition 2 *(Lower bound) For any dag G of order n,*

$$w^*(G) \geq \log \mu(G) - \log n.$$

Proof: Fix a dag G of order n. Let μ be the representation complexity of G. By the averaging argument, there must exist at least one vertex with $\log(\mu/n)$ parents; therefore, the moralized G has a clique of size at least $\log(\mu/n) + 1$, which proves the proposition. ∎

We show that the above bound is tight for small μ in the sense that there exist dags that come arbitrarily close to it.

Proposition 3 *For each integer $c \geq 1$, there exists a graph G_c with $w^*(G_c) = \frac{1}{c}\log n$ such that $\log \frac{\mu(G_c)}{n} = \log(n^{1/c} - O(\frac{\log n}{n^{1-1/c}}))$.*

Proof: Fix $c \geq 1$. Let G_c be the complete bipartite graph with the smaller component of size $\frac{1}{c}\log n$ such that all the edges are directed from the smaller component to the larger. It is easy to see that $w^*(G_c) = \frac{1}{c}\log n$. On the other hand, $\mu(G_c) = \frac{1}{c}\log n + n^{1/c}(n - \frac{1}{c}\log n)$. The claim easily follows. ∎

Notice that the bound is not tight up to a $\log n$ additive term as $\mu(K_n) = 2^n - 1$, and $w^*(K_n) = n - 1$ (where K_n denotes clique on n nodes), while the bound gives $w^*(K_n) > n - \log n$. The claim follows from the convexity of the bound.

Proposition 4 *(Upper bound) For any dag G of order n, $w^*(G) \leq \lfloor \log(\mu(G) - 2(n+1)) \rfloor + 1$.*

Proof: Omitted (see [3]). ∎

4 Testing the treewidth

In this section we present some observations on how to quickly bound the treewidth during learning. We do not attempt to approximate the treewidth (let alone compute it exactly); this would be an overkill in this setting. We consider a more relaxed task of testing whether the treewidth is "small enough". The tests will be allowed to err with small probability, provided they are very fast (sublinear in the number of nodes in the network). This would allow us to choose between alternative networks without actually incurring the expense of running good (but much more expensive) approximation algorithms for determining the treewidth.

Let $\bar{G} = (V, \bar{E})$ denote the complement (to a complete graph) of an undirected graph $G = (V, E)$, i.e. $\bar{E} = S$

E where S is the set of all possible edges among the nodes in V. Let us also use the notation $w^*(G,\bar{G}) = w^*(G) + w^*(\bar{G})$. First, we show that for any graph G, either G or its complement $\bar{G}$ must have large treewidth, i.e. that $w^*(G,\bar{G})$ is $\Theta(n)$, where $n = |V|$.

Proposition 5 *For any undirected graph G, $w^*(G,\bar{G}) \geq (2-\sqrt{3})n$.*

Proof: See Appendix. ∎

The proof of the proposition (see Appendix) suggests that any two-sided error probabilistic algorithm for testing whether a given graph G has treewidth at most some fixed integer k, first approximates the density of G by randomly selecting a small set of vertex pairs and outputting the fraction of them that are adjacent in the graph. By the Chernoff bound, a sample of size m is sufficient to get within an additive error of δ with probability at least $\exp(-2\delta^2 m)$; thus setting $m = \delta^{-2}$ upper bounds the probability of error by $e^{-2} < 1/7$. If the estimate is less than $nr - \binom{r+1}{2}$ (as follows from the proof), we reject. Thus we may assume that G has $O(nr)$ edges, i.e. that G is sparse.

Since most learning algorithms are incremental, i.e. based on local edge transformations, we only need to provide incremental treewidth checks; in other words, we only need to make sure that such local modifications do not inflict large increases in the treewidth. Clearly, no edges should be added to a node, which already has more than r parents (since moralization will induced a clique of size more than $r+1$). Notice that this degree bound is necessary, but not sufficient (recall the grid example). It is not hard to see that adding a single directed edge cannot increase the treewidth of the resulting moral graph by more than one (see Appendix). Furthermore, it turns out that *most* one-edge extensions do not increase the treewidth for undirected graphs. Figure 3(b) shows the statistics for the average increase in treewidth over all one-edge extensions, where x-axis corresponds to the treewidth of the average extension, and y-axis gives the number of undirected graphs whose average extension has the corresponding treewidth.

5 On the distribution of representation complexity

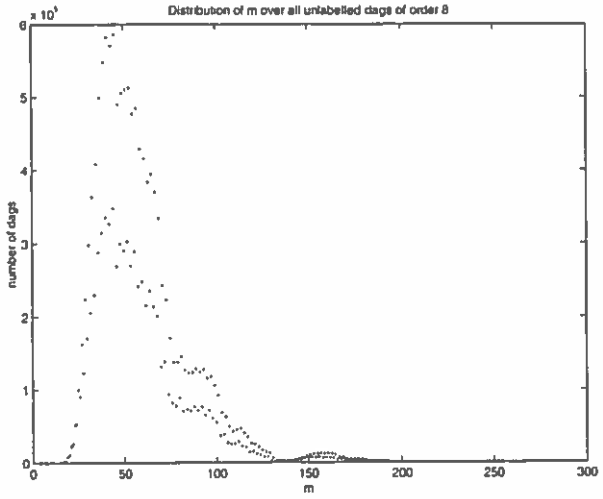
We will now examine how $\mu(\cdot)$ is distributed across all dags on a given number of nodes (see Figure 3). Our purpose here is to explain the behavior of the distribution and to give a feeling for the asymptotics, rather than to give rigorous estimates; very crude bounds are sufficient for the argument.

It is prominant that most dags are concentrated on a small number of μ values. Consider a labeled n-node dag G. We can associate G with a linear ordering of nodes $1,\cdots,n$ in a natural way (so that there is no edge from a higher- to a lower-numbered node). Let d_i denote the in-degree of node i; notice that for all i, $d_i \leq i-1$. We have $\mu(G) = \sum_{i=1}^{n} 2^{d_i}$. What values of μ are shared by many dags? Clearly the multiplicity of μ depends on the number of different partitions of μ, i.e. ordered sequences of in-degrees $(d_1,\ldots,d_n)$ satisfying $\sum_{i=1}^{n} 2^{d_i} = \mu$ and $d_i \leq i-1$ for all i. The multiplicity of μ is the sum over all such partitions of the number of labeled dags with this in-degree sequence. Clearly, the number of ordered labelled dags with a given sequence $(d_1,\ldots,d_n)$ is bounded by $\binom{n-1}{d_n}\binom{n-2}{d_{n-1}}\cdots\binom{1}{d_2}$, which is maximized[5] when $d_i = \left\lceil \frac{i-1}{2} \right\rceil$ for all i. (This results in dags with $\lfloor \frac{n}{2} \rfloor \lceil \frac{n}{2} \rceil$ edges.) So what values of μ correspond to this choice of d_i's? We have $\mu = \sum_{i=1}^{n} 2^{\lceil \frac{n-i}{2} \rceil} \leq 3 \cdot 2^{n/2}$. Indeed, the values around $3 \cdot 2^{\frac{n}{2}}$ correspond to the sharp peak in the distribution of μ. This is already true for small values of n. For example, for $n = 8$ the peak is around $3 \cdot 2^{\frac{8}{2}} = 48$ (see Figure 3a).

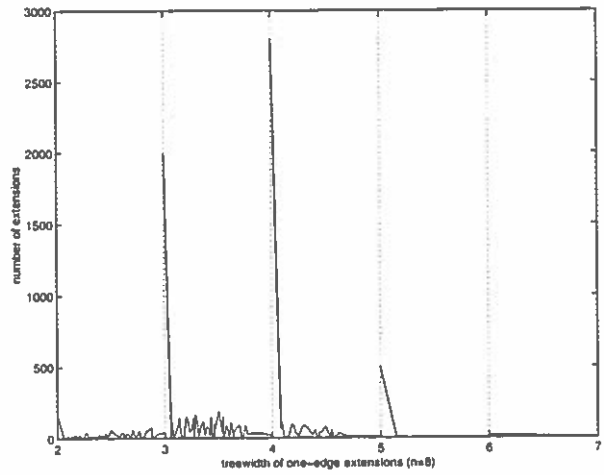
Also notice that there are two self-similar curves in Figure 3a, one above the other, corresponding to odd and even values of $\mu(\cdot)$ respectively. This can be easily explained by observing that μ of a dag is odd exactly when the dag has an odd number of sources (i.e. vertices with in-degree 0). Let $p(n,k)$ be the probability that a random n-node dag has k sources. The limits of $p(n,k)$ were determined by Liskovec [16]. Roughly, $\frac{p(n,k-1)}{p(n,k)} \sim \frac{k2^{k-1}}{\rho A(k)}$, where $\rho \approx 1.488$ and $A(k)$ is a decreasing function in k with values in $(1,2)$. Clearly the ratio grows exponentially with k. We have $A(2) \approx 1.68$, hence the ratio of the number of dags with a single source to the number of dags with two sources is $\frac{4}{\rho A(2)} \approx 1.6$, which explains why there are more dags with odd number of parameters.

Notice a sharp decrease of the number of dags after μ becomes larger than 2^{n-1}. This is due to the fact that all dags with at least 2^{n-1} parameters must have a node of in-degree at least $(n-2)$ (since there can be at most three nodes with in-degree $(n-3)$, and even if the other $(n-3)$ nodes form a clique, the number of parameters can be at most $3 \cdot 2^{n-3} + 2^{n-3} - 1 = 2^{n-1} - 1$). It is not hard to see that the number of such dags is of the same order as the number of dags on

[5] This number is approximately $\frac{\prod_{i=1}^{n-1} 2^i}{\sqrt{(n-1)!}} \sim 2^{\frac{n^2 - n\log n}{2}}$



(a) $n = 8$



(b) $n = 8$

Figure 3: (a) The distribution of dags according to $\mu(\cdot)$. (b) Average increase in treewidth over all one-edge extensions.

$(n-1)$ nodes. Stanley [24] gave an asymptotic formula for the number of labeled n-node dags, yielding the ratio $\sim \frac{n2^{n-1}}{e}$ for the number of dags on n nodes with at least 2^{n-1} parameters to the total number of n-node dags; hence the small bump to the right of the main peak is just an exponentially scaled-down version of this peak for $(n-1)$-node dags.

It was shown in [14] that almost every n-node graph with at least $1.18n$ edges has a large treewidth, i.e. has treewidth $\Theta(n)$. Using this fact, it is easy to see that there are dags that represent a whole spectrum of different treewidths, yet all have the same representation complexity. For example, consider the dags with $\mu = n^k$ parameters for some $k \geq 1$. Recall that $\mu = \sum_{i=1}^{n} 2^{d_i}$, where d_is denote the indegrees. Clearly, for all i, $d_i \leq k \log n$, hence the number of edges in such graphs is at most $O(n \log n)$. Now the claim follows. Indeed, a. e. graph with the number of edges between $1.18n$ and $O(n \log n)$ has treewidth $\Theta(n)$. On the other hand, it is not hard to construct graphs with $< 1.18n$ edges that have the same μ and constant treewidth; for example, any graph that does not contain K_4 (a clique on 4 nodes) as a minor has treewidth 2, and there certainly are K_4-free graphs with $< 1.18n$ edges.

6 Discussion and Conclusions

Traditional model-selection criteria used for learning Bayesian networks favor models that fit the data and have low representation complexity (i.e. the number of parameters needed to specify the network). However, those criteria do not take into account the complexity of inference in the resulting network, which is exponential in the network's treewidth. Herein, we have demonstrated the importance of using the treewidth as a part of model selection criteria. We showed that it can frequently occur that two networks have drastically different treewidths, and yet represent close distributions (in terms of KL-divergence) and have similar representation complexity (and thus similar BIC/MDL scores). We also suggest some efficient probabilistic tests for a quick estimation of the treewidth during learning.

Clearly, our current work is an initial theoretical investigation that requires further empirical testing, both on artificially simulated benchmarks and on practical problems. In our future work, we plan an in-depth investigation of certain realistic applications that can benefit from our approach, i.e. applications that can be successfully approximated by probabilistic models supporting simple inference, versus those applications that are inherently "non-approximable".

Finally, we provide a high-level discussion of the "approximability" issues which may give some intuition behind our approach and explain our belief in its practical success. It is important to note that the treewidth is the *worst-case* measure of the inference complexity, because it does not take into account the numerical parameters of the distribution: the treewidth of all networks encoding the same set of independence assumptions (i.e. probability distributions consistent with a given graph structure) is the same. However, some subsets of nodes may be almost independent while others may yield nearly-deterministic dependencies. Some recent studies suggest that "local" approximation techniques imposing additional independence assumptions (e.g., mini-bucket approximations [8, 20] for finding most-probable diagnosis, naive Bayes for classification [19], or (generalized) iterative belief propagation algorithms for probabilistic decoding [10, 25]) work surprisingly well in presence of such nearly-deterministic dependencies (see [20, 19] for initial theoretical results). A further theoretical understanding of this phenomenon is a direction for our future work.

In practice, there are indeed many problems that give rise to such "unbalanced" distributions, that combine very weak dependencies with very strong ones. One example relates to performance management in distributed computer systems [12]: recognizing "high-level" transactions from "low-level" commands in order to model and predict the user's behavior can be viewed as a probabilistic classification problem. The real-life data often exhibit either nearly-deterministic dependencies or almost no dependence among certain sets of transactions, which is not that surprising given the nature of data produced by a deterministic source code under some limited variation of the input conditions and the environment. Another example of this sort is problem determination and fault isolation in distributed computer systems using a Bayesian network to model the dependencies between certain software or hardware faults and various system events (alarms, messages) [20]. Yet another example relates to the area of noisy channel coding (e.g., Turbo-codes and Low-Density-Parity-Check codes), which mainly involves deterministic (encoding) and nearly-deterministic (low-noise transmission) dependencies, and yields superb performance of local approximations, as noted above [10, 13, 9].

In summary, we wish to emphasize that the strength of probabilistic dependencies between the nodes may be quite different for different distributions among those represented by the same graph structure G (we denote the set of all such distributions by Ω_G), and thus certain distributions in Ω_G may allow better approximations by small-treewidth graph structures. A parameter more refined than the graph-based treewidth is needed that would allow to distinguish among such distributions. Furthermore, we often have some knowledge about the distribution of queries to be posed, which also contributes to the *effective* inference complexity of the model. For example, large cliques in a graph might be mostly due to dependencies that are either weak or irrelevant for answering a query chosen according to our query distribution; hence the complexity of inference should be measured *with respect to this query distribution.* These and related questions are yet another avenue of our further research.

References

[1] E. Amir. Efficient approximation for triangulation of minimum treewidth. In *17th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2001.

[2] A. Beygelzimer and I. Rish. On the importance of using treewidth as a model-selection criterion for learning Bayesian networks. In *In 7th Valencia international meeting on Baysian statistics*, 2002.

[3] Alina Beygelzimer and Irina Rish. Learning Bayesian networks that are efficient for inference. Technical report, IBM T.J. Watson Research Center, 2001.

[4] G.F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2–3):393–405, 1990.

[5] B. D'Ambrosio. Symbolic probabilistic inference in large BN2O networks. In *Proc. Tenth Conf. on Uncertainty in Artificial Intelligence*, pages 128–135, 1994.

[6] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Proc. Twelfth Conf. on Uncertainty in Artificial Intelligence*, pages 211–219, 1996.

[7] R. Dechter. *Bucket elimination: A unifying framework for probabilistic reasoning.* In M. I. Jordan (Ed.), Learning in Graphical Models, Kluwer Academic Press, 1998.

[8] R. Dechter and I. Rish. Mini-buckets: A General Scheme for Approximating Inference. *To appear in J. of ACM*, 2002.

[9] B.J. Frey. *Graphical models for machine learning and digital communication*. MIT Press: Cambridge, MA, 1998.

[10] B.J. Frey and D.J.C. MacKay. A revolution: Belief propagation in graphs with cycles. *Advances in Neural Information Processing Systems*, 10, 1998.

[11] D. Heckerman. A tutorial on learning Bayesian networks, technical report msr-tr-95-06. Technical report, Microsoft Research, 1995.

[12] J. Hellerstein, Jayram Thathachar, and I. Rish. Recognizing end-user transactions in performance management. In *Proceedings of AAAI-2000*, pages 596–602, Austin, Texas, 2000.

[13] K. Kask I. Rish and R. Dechter. Approximation algorithms for probabilistic decoding. In *Uncertainty in Artificial Intelligence (UAI-98)*, 1998.

[14] T. Kloks. *Treewidth. Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1994.

[15] S.L. Lauritzen and D.J. Spiegelhalter. Local computation with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, 50(2):157–224, 1988.

[16] V. A. Liskovec. The number of maximal vertices of a random acyclic digraph. *Theory Prob. Applic.*, 20:401–409, 1975.

[17] D.J.C. MacKay and R.M. Neal. Near Shannon limit performance of low density parity check codes. *Electronic Letters*, 33:457–458, 1996.

[18] Ann E. Nicholson and Nathalie Jitnah. Using mutual information to determine relevance in bayesian networks. In *Pacific Rim International Conference on Artificial Intelligence*, pages 399–410, 1998.

[19] I. Rish. An empirical study of the naive Bayes classifier. In *Proceedings of IJCAI-2001 workshop on Empirical Methods in AI*, Seattle, WA, 2001.

[20] Irina Rish, Mark Brodie, and Sheng Ma. Accuracy versus efficiency in probabilistic diagnosis. Technical report, IBM T.J. Watson Research Center, 2002.

[21] Irina Rish, Mark Brodie, and Sheng Ma. Intelligent probing: a Cost-Efficient Approach to Fault Diagnosis in Computer Networks. *Submitted to IBM Systems Journal*, 2002.

[22] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[23] M. Shwe, B.F. Middleton, D.E. Heckerman, M. Henrion, E.J. Horvitz, H. Lehmann, and G.F. Cooper. Probabilistic diagnosis using a reformulation of the Internist-1/QMR knowledge base: I. The probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30:241 – 255, 1991.

[24] R. Stanley. Acyclic orientations of graphs. *Discrete Mathematics*, 5:171–178, 1973.

[25] J. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *NIPS 13*, pages 689–695. MIT Press, 2001.

[26] N.L. Zhang and D. Poole. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328, 1996.

Appendix

Proposition 1 [3, 2] *For every $n > 3$, there exist dags $G = G(n)$ and $G' = G'(n)$ such that $\mu(G) > \mu(G')$, whereas $w^*(G) = 2$, while $w^*(G') = \sqrt{n}$.*

Proof: Let G be the complete bipartite graph with the "top" component of size 2 such that the direction of all edges is from this component to the "bottom" component of size $n-2$. Let G' be the $\sqrt{n} \times \sqrt{n}$ rectangular grid. (Since the proposition is of asymptotic nature, we ignore the issue of divisibility.) Let the direction of all vertical (resp. horizontal) edges in G' be from left to right (resp. from top to bottom). Clearly, the moral graph of G has treewidth 2 (note that the moral graph is already triangulated), while the treewidth of the moralized G' is $\sqrt{n}$. We only need to show that $\mu(G) \sim \mu(G')$. Indeed, $\mu(G) = 4(n-2) + 2$ (4 parameters for each of the $(n-2)$ nodes in the bottom level, and one prior for each top-level node), while $\mu(G') = 4(\sqrt{n}-1)^2 + 4(\sqrt{n}-1) + 1 = 4(n - \sqrt{n}) + 1$. Thus in fact $\mu(G') < \mu(G)$. Clearly the two quantities can be made equal by adding additional edges to G', which can only increase its treewidth. ∎

Lemma 1 *With probability at least $1 - exp(-\frac{ld^2}{r^2} + 2\ln r)$, a random (l, r, d)-diffuser has treewidth $r - 1$.*

Proof: It is easy to show that the treewidth does not exceed $r-1$, i.e. one less than the size of the clique on

the parent (i.e., right) component. Therefore, it sufficies to show that the moralization can indeed induce such a clique with the required probability. Let L and R denote the left and the right components respectively. What is the probability that a pair of parents $p, q \in R$ is not connected in the moralized graph?

$$\begin{aligned}
&\Pr[p \text{ and } q \text{ are not connected}] \\
&\quad = \Pr[\forall v \in L:\ p \notin \Gamma(v) \vee q \notin \Gamma(v)] \\
&\quad = (1 - d^2/r^2)^l \\
&\quad \leq e^{-ld^2/r^2}.
\end{aligned}$$

Let P be the expected number of pairs in R that are not induced. Then $\mathbf{E}[P] = \binom{r}{2} e^{-ld^2/r^2}$, and by Markov's inequality, the probability that there are no more than $k\mathbf{E}[P]$ such edges is at least $1 - 1/k$. Thus we want the largest k such that $k\mathbf{E}[P] < 1$, so $k \sim exp(\frac{ld^2}{r^2} - 2\ln r)$, proving the lemma. ∎

Proposition 5 *For any undirected graph G, $w^*(G, \bar{G}) \geq (2 - \sqrt{3})n$.*

Proof: Let $e(G)$ and $r = w^*(G)$ denote the number of edges and the treewidth of a graph $G = (V, E)$, respectively. It is easy to see that $e(G) \leq nr - \frac{1}{2}r(r+1)$. Indeed, consider the ordering of V minimizing the treewidth. Every vertex in V is adjacent to at most r higher ordered vertices except for the last r vertices, which, in the worst case, form a clique. Thus we have $e(G) \leq \sum_{i=1}^{n-r} r + \binom{r}{2} = nr - \frac{1}{2}r(r+1)$, proving the claim. We need to show that $w^*(\bar{G}) \geq (2 - \sqrt{3})n - r$. Assume the contrary; then a similar argument shows that $e(\bar{G}) \leq n((2 - \sqrt{3})n - r - 1) - \frac{1}{2}((2 - \sqrt{3})n - r)((2 - \sqrt{3})n - r - 1)$. This, however, contradicts $e(\bar{G}) = n(n-1)/2 - e(G) \geq n(n-1)/2 - nr + \frac{1}{2}r(r+1)$. Indeed, we merely need to verify that this lower bound is greater than the upper bound given by the assumption on $w^*(\bar{G})$, which reduces to showing that $2r(r+1) + c^2 + c > 2r(n-c)$ holds for all $0 < r < n - c$, where $c = (\sqrt{3} - 1)n$. We have $r \leq n - c - 1$; otherwise the proposition trivially holds. Therefore it sufficies to show that $(\sqrt{3} - 1)^2 n^2 > 2(2 - \sqrt{3})^2 n^2$, which is certainly true. ∎

Proposition 6 *For any dag G, adding a single directed edge to G cannot increase $w^*(G)$ by more than one.*

Proof: Indeed, if G has treewidth w, then there must exist an ordering of vertices $o = v_1, \ldots, v_n$ such that for each i, the width of v_i is at most w. Assume that we add a directed edge $v_i \to v_j$ (clearly it should not create a directed cycle in G). The moralization of the augmented graph induces a clique on $\Gamma(v_j) \cup \{v_i\}$. We will construct an ordering of this moral graph with width at most $w + 1$. In fact, we only need to move v_i to the end of o. Moving a node downward in the ordering can only decrease its width. The width of v_j and the width of every node in $\Gamma(v_j)$ will increase by one. Clearly, the width of any other vertex with index smaller than i in o, will not change. The only worry are the neighbors of v_i with index greater than i that have now percolated up, and thus will be eliminated before v_i during the inference. However, all that can happen in the worst case, is that every such node will get connected to v_i, increasing its width by one. Thus the width of every node can increase by at most one. ∎

On the logic of d-separation

Balder ten Cate
Institute for logic, language and computation
University of Amsterdam
Nieuwe Doelenstraat 15
1012CP Amsterdam
b.ten.cate@hum.uva.nl

Abstract

D-separation is a key notion from the belief networks literature. Two logics are introduced for reasoning about d-separation in directed graphs, and it is shown how the sentences of these logics can be translated into Converse Propositional Dynamic Logic. Various consequences of this result are discussed regarding model checking, satisfiability checking and the relation between d-separation and conditional independence in probability distributions.

1 Introduction

Belief networks (Pearl, 2000), also called *Bayesian networks*, constitute a formalism for efficiently representing and reasoning with probability distributions. In general, probability distributions can be very large: exponentially large in the number of random variables. However, in practice there is a lot of redundancy in the data, due to independencies between random variables. By explicitly representing the dependencies, the probability distribution can be represented more efficiently. For example, consider the probability distribution given in Table 1. In this probability distribution, all three random variables happen to be independent (for example, the chance of winning the lottery given that it rains, is the same as the chance of winning the lottery given that it doesn't rain). By exploiting this independence, a more succinct representation is possible, as in Table 2.

Table 1: Example probability distribution

rain	paper_accepted	win_lottery	P
0	0	0	0.18
0	0	1	0.02
0	1	0	0.18
0	1	1	0.02
1	0	0	0.27
1	0	1	0.03
1	1	0	0.27
1	1	1	0.03

Table 2: Compact description of Table 1

- $P(\texttt{rain}) = 0.6$
- $P(\texttt{paper\_accepted}) = 0.5$
- $P(\texttt{win\_lottery}) = 0.1$
- `rain`, `paper_accepted` and `win_lottery` are independent.

This suggests the following method for efficiently representing probability distributions: first partition the set of random variables into independent subsets. Then construct the probability distributions for these separate sets of random variables. Although this method works, it is still not the optimal way of exploiting independence: merely partitioning the set of random variables is a very rough way of encoding independencies. A more fine-grained treatment of independence makes it possible to represent probability distributions even more efficiently. Belief networks constitute one such representation.

In belief networks, the independence between random variables is represented by means of a *directed graph*. The nodes of the graph are the random variables themselves, and the edges indicate direct dependencies between them. An example belief network is depicted in Figure 1.

Formally, a belief network consists of two parts: (i) a finite, acyclic directed graph, and (ii) a function assigning probability tables to the nodes of the graph. In other words, it is a structure (V, R, π) where V is

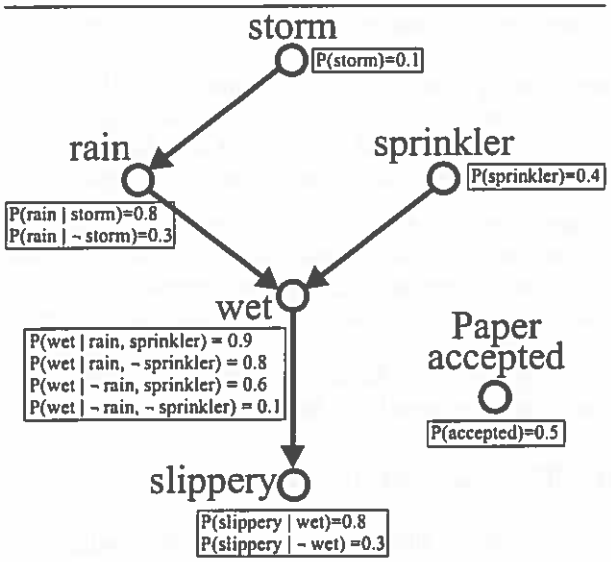


Figure 1: Example belief network

a finite set of nodes (the *random variables* of the belief network), R is an acyclic relation over V, and π is a function assigning conditional probability tables to the elements of V.

In this paper, we are not interested in the numerical probabilities assigned by π. Rather, we will investigate the underlying directed graph (V, R) of the belief network, which represents the independencies between the random variables. In particular, we will focus on the formal properties of the notion of *d-separation*. D-separation tells us how to interpret the graph: whereas the edges of the graph represent direct (causal) dependencies between the random variables, *d-separation* computes from these the *indirect* dependencies.

2 D-separation

D-separation tells us when two (sets of) random variables, i.e., nodes in the graph, are dependent. More precisely, d-separation is a ternary relation between sets of nodes. With $(X \perp\!\!\!\perp Y|Z)$ we will denote that X and Y are independent given (*d-separated by*) Z. Formally, d-separation is defined as follows (Pearl, 1988).[1]

Let $\mathcal{G} = (V, R)$ be a directed graph. An undirected path in $\mathcal{G}$ is a sequence of nodes and arrows $x_1 a_1 \cdots a_{n-1} x_n$ $(n \geq 1)$ where $x_1, \ldots, x_n \in V$ and $a_1, \ldots, a_{n-1} \in \{\leftarrow, \rightarrow\}$, such that whenever $x_i \rightarrow x_j$ or $x_j \leftarrow x_i$ occurs in the sequence, $(x_i, x_j) \in R$.For example, the sequence *storm* $\rightarrow$ *rain* $\rightarrow$ *wet* $\leftarrow$ *sprinkler* is a valid path in the example depicted in Figure 1.

Definition 1 (Blocking) *An undirected path $x \ldots y$ is* blocked *by a set of nodes Z if at least one of the following conditions hold.*

1. $x \in Z$ *or* $y \in Z$
2. $\cdots \rightarrow z \rightarrow \cdots$ *occurs on the path, for some* $z \in Z$
3. $\cdots \leftarrow z \leftarrow \cdots$ *occurs on the path, for some* $z \in Z$
4. $\cdots \leftarrow z \rightarrow \cdots$ *occurs on the path, for some* $z \in Z$
5. $\cdots \rightarrow v \leftarrow \cdots$ *occurs on the path, and neither v nor any of its descendants is in Z.*

Definition 2 (D-separation) *Let $\mathcal{G} = (V, R)$ and $X, Y, Z \subseteq V$. Z d-separates X from Y, $(X \perp\!\!\!\perp Y|Z)_{\mathcal{G}}$, iff every undirected path from a node in X to a node in Y is blocked by Z.*

Intuitively, the paths that are not blocked by Z corresponds to "lines of reasoning", and Z itself corresponds to the prior information. Some examples of d-separation will illustrate this. Consider again the belief network depicted in Figure 1. In this network, it holds that $(\{\texttt{rain}\} \perp\!\!\!\perp \{\texttt{paper\_accepted}\}|\emptyset)$, which corresponds to our intuition: the two random variables are considered independent. One should realise that, for two variables to be dependent, it doesn't suffice that they have a common effect. In the example it holds that $(\{\texttt{rain}\} \perp\!\!\!\perp \{\texttt{sprinkler}\}|\emptyset)$. However, suppose that we have prior information about whether it is wet. Then, conditional on this, *rain* and *sprinkler* need not be independent anymore ("I already knew that it is wet. Now I learn that it didn't rain. I conclude from this that the sprinkler was on"). Indeed, $(\{\texttt{rain}\} \perp\!\!\!\perp \{\texttt{sprinkler}\}|\{\texttt{wet}\})$ is *false* in the belief network.

The definition of d-separation given above is phrased in terms of *blocked paths*. Alternatively, one can give a definition based on paths that are *not* blocked. Such a definition can be given in terms of finite state automata, relying on the fact that paths in the graph that are not blocked by Z constitute a regular set.

Definition 3 (D-separation automaton) *The* d-separation automaton for Z *is the nondeterministic finite state automaton depicted in Figure 2. The automaton starts in the* s *state, at any point in the underlying graph of the belief network. During execution,*

[1] In the definition given by Pearl, $(X \perp\!\!\!\perp Y|Z)$ is undefined if X, Y and Z are not disjoint. However, for present purposes, it is more convenient to consider arbitrary sets.

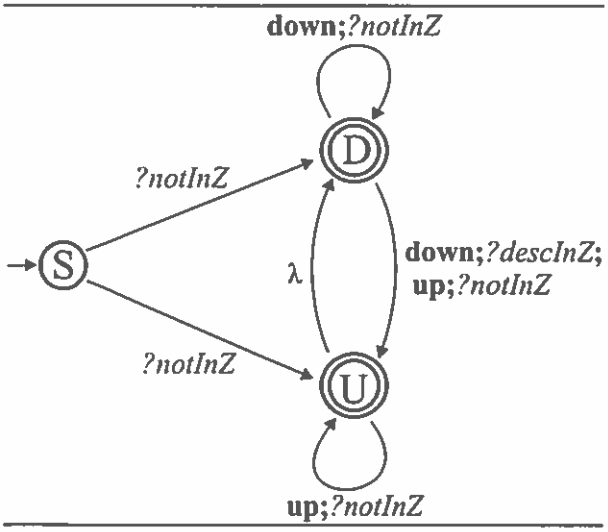


Figure 2: The d-separation automaton

it can "walk through" the belief network, by executing any of the following four actions.

up	*Make one step in the graph, against the direction of the arrows*
down	*Make one step in the graph, in the direction of the arrows*
?notInZ	*Check that the current node is not an element of Z*
?descInZ	*Check that there is a directed path (possibly of length 0) from the current node to an element of Z*

The λ in Figure 2 means skip *("don't do anything"). The automaton can terminate whenever it is in the* U *or in the* D *state.*

It can easily be seen that the succesful runs of the automaton corresponds precisely to the paths in the graph that are not blocked by Z. In other words, the automaton constitutes an alternative, equivalent definition of d-separation: X and Y are d-separated by Z if there is no succesful run of the d-separation automaton, starting at a node in X and ending at a node in Y.

D-separation is closely related to conditional independency in probability distributions (Geiger and Pearl, 1990). However, the exact relationship between d-separation and conditional independence hasn't been established yet.

Table 3 shows a list of properties of d-separation discussed by Pearl (1988), where $X, Y, Z, \alpha, \beta, \ldots \subseteq V$ are assumed to be disjoint and nonempty, and $\alpha, \beta, \ldots$ are singletons. Of these properties, symmetry, composition, weak union and contraction are also valid for conditional independence in probability distributions.

In the present paper, two logics for reasoning about d-separation are introduced, and it is shown that the formulas of these languages can be translated into certain modal logics. Various consequences of this result are discussed regarding model checking, satisfiability checking and the relation between d-separation and conditional independence in probability distributions.

3 The logic of d-separation $\mathcal{LD}$

Consider again the list of properties of d-separation given in Table 3. While Pearl did not explicitly restrict himself to a particular language for formulating these properties, in this paper, the language is taken more seriously. In the present section, the logic of d-separation $\mathcal{LD}$ is introduced, and in the next section, this logic is extended by adding *singleton variables*.

3.1 Syntax and semantics of $\mathcal{LD}$

The language of $\mathcal{LD}$ is built up from basic propositions such as $(A \perp\!\!\!\perp B|C)$ or $(A = B)$, using the boolean connectives $\neg, \wedge, \vee$ and $\rightarrow$. The terms $A, B, C, \ldots$ themselves also have internal structure: they are built up from variables $X, Y, Z, \ldots$ using the boolean connectives $-, \cap$ and $\cup$ (cf. Table 4).

It is important to see the difference between the variables of the language, and the random variables in terms of which the belief network is defined. As we are only interested in the graphical structure of the belief network, we will henceforth refer to the random variables of the network as *nodes*. The variables that occur in the language, which we will simply refer to as *variables*, will be interpreted as referring to sets of nodes.

The basic propositions of the form $A = B$ will be interpreted as identity statements. It is convenient to have them in the language, because it allows us to express disjointness of variables.[2] As a matter of fact, it is not strictly necessary to explicitly introduce identity statements in the language, since identity can already

[2]Recall that d-separation was defined for arbitrary sets of nodes, instead of requiring disjointness from the start (as does Pearl (1988)). It should be clear to the reader at this point, that this choice was not harmful, since the language of $\mathcal{LD}$ is rich enough to express disjointness of sets of nodes.

Table 3: Some properties of d-separation

1. Symmetry.	$(X \perp\!\!\!\perp Y \mid Z) \leftrightarrow (Y \perp\!\!\!\perp X \mid Z)$
2. (De)composition.	$(X \perp\!\!\!\perp Y \cup W \mid Z) \leftrightarrow (X \perp\!\!\!\perp Y \mid Z) \wedge (X \perp\!\!\!\perp W \mid Z)$
3. Intersection.	$(X \perp\!\!\!\perp Y \mid Z \cup W) \wedge (X \perp\!\!\!\perp W \mid Z \cup Y) \rightarrow (X \perp\!\!\!\perp Y \cup W \mid Z)$
4. Weak union.	$(X \perp\!\!\!\perp Y \cup W \mid Z) \rightarrow (X \perp\!\!\!\perp Y \mid Z \cup W)$
5. Contraction.	$(X \perp\!\!\!\perp W \mid Z \cup Y) \wedge (X \perp\!\!\!\perp Y \mid Z) \rightarrow (X \perp\!\!\!\perp Y \cup W \mid Z)$
6. Weak transitivity.	$(X \perp\!\!\!\perp Y \mid Z) \wedge (X \perp\!\!\!\perp Y \mid Z \cup \gamma) \rightarrow (X \perp\!\!\!\perp \gamma \mid Z) \vee (\gamma \perp\!\!\!\perp Y \mid Z)$
7. Chordiality.	$(\alpha \perp\!\!\!\perp \beta \mid \gamma \cup \delta) \wedge (\gamma \perp\!\!\!\perp \delta \mid \alpha \cup \beta) \rightarrow (\alpha \perp\!\!\!\perp \beta \mid \gamma) \vee (\alpha \perp\!\!\!\perp \beta \mid \delta)$

Table 4: Language of $\mathcal{LD}$

$$\begin{array}{rcl} \textit{Terms} & ::= & X \mid \emptyset \mid -A \mid A \cap B \mid A \cup B \\ \textit{Statements} & ::= & (A \perp\!\!\!\perp B \mid C) \mid A = B \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \end{array}$$

Table 5: Semantics of $\mathcal{LD}$

$$\begin{array}{rcl} X^{\mathcal{M}} & = & I(X) \\ \emptyset^{\mathcal{M}} & = & \emptyset \\ (-A)^{\mathcal{M}} & = & V \backslash A^{\mathcal{M}} \\ (A \cap B)^{\mathcal{M}} & = & A^{\mathcal{M}} \cap B^{\mathcal{M}} \\ (A \cup B)^{\mathcal{M}} & = & A^{\mathcal{M}} \cup B^{\mathcal{M}} \\ \mathcal{M} \models (A \perp\!\!\!\perp B \mid C) & \text{iff} & C^{\mathcal{M}} \text{ d-separates } A^{\mathcal{M}} \\ & & \text{from } B^{\mathcal{M}} \text{ in } (V, R) \\ \mathcal{M} \models (A = B) & \text{iff} & A^{\mathcal{M}} = B^{\mathcal{M}} \\ \mathcal{M} \models \neg\phi & \text{iff} & \mathcal{M} \not\models \phi \\ \mathcal{M} \models \phi \wedge \psi & \text{iff} & \mathcal{M} \models \phi \text{ and } \mathcal{M} \models \psi \\ \mathcal{M} \models \phi \vee \psi & \text{iff} & \mathcal{M} \models \phi \text{ or } \mathcal{M} \models \psi \\ \mathcal{M} \models \phi \rightarrow \psi & \text{iff} & \mathcal{M} \not\models \phi \text{ or } \mathcal{M} \models \psi \end{array}$$

be expressed in terms of d-separation: $A = B$ and $(A \perp\!\!\!\perp A \mid B) \wedge (B \perp\!\!\!\perp B \mid A)$ are equivalent.

A model for this language consists of a (non-empty) directed graph (V, R), together with an interpretation function I that assigns sets of nodes of the graph to the variables. Of course, for the models to be the underlying graph of a belief network, they have to be finite and acyclic, but as we will see in the next section, there is no need to restrict ourselves to this particular class of models.

Formally, the semantics of $\mathcal{LD}$ is given in Table 5, where $\mathcal{M} = (V, R, I)$.

Recall that we defined a model for our language to be a directed graph, together with an interpretation function. Of course, the idea is that the models represent the underlying graphs of belief networks. However, for that to be the case, they have to be finite and acyclic. In the next section we will show that such a restriction on the class of considered models, is not necessary. More precisely, we will see that $\mathcal{LD}$ has the finite, acyclic, weakly connected model property, which means that if a formula is satisfiable, then it is also satisfiable in a finite, acyclic, weakly-connected model.

3.2 $\mathcal{LD}$ has the finite, acyclic, weakly connected model property

In this section, it is proven that any satisfiable sentence of $\mathcal{LD}$, is satisfied in a finite, acyclic, weakly connected model. A model is weakly connected, if there is an undirected path between every two points. We start by introducing three basic model-theoretic notions.

Definition 4 *$(V, R, I) \subseteq (V', R', I')$ if $V' \subseteq V$, $R' \subseteq R$ and $I' = I|_{V'}$ (i.e., the restriction of I to V').*

Notice that this is *not* the standard notion of a submodel. Usually, $\mathcal{N}$ is considered a submodel of $\mathcal{M}$ if $\mathcal{N}$ can be obtained from $\mathcal{M}$ by removing some nodes (plus their incoming and outgoing arrows). As we have defined it here, $\mathcal{N} \subseteq \mathcal{M}$ means that $\mathcal{N}$ can be obtained from $\mathcal{M}$ by removing some nodes *and/or arrows*.

Definition 5 *Let $\mathfrak{C}$ be a nonempty set of models. The disjoint union (or* direct sum*) $\oplus\mathfrak{C}$ is $(\mathbf{V}, \mathbf{R}, \mathbf{I})$, where*

$$\begin{array}{lcl} \mathbf{V} & = & \{(\mathcal{M}, v) \mid \mathcal{M} \in \mathfrak{C} \textit{ and } v \in V^{\mathcal{M}}\} \\ (\mathcal{M}, v)\mathbf{R}(\mathcal{N}, w) & \textit{iff} & \mathcal{M} = \mathcal{N} \textit{ and } (v, w) \in R^{\mathcal{M}} \\ (\mathcal{M}, v) \in \mathbf{I}(X) & \textit{iff} & v \in I^{\mathcal{M}}(X) \end{array}$$

(where $V^{\mathcal{M}}$ denotes the domain of $\mathcal{M}$, etc.)

The third notion that we will introduce, is that of weakly connected components.

Definition 6 *Let $\mathcal{M} = (V, R, I)$. A nonempty subset $V' \subseteq V$ is called a* weakly connected component *if (1) every two points in V' are connected by an undirected path in $\mathcal{M}$, and (2) there is no strict superset of V' satisfying (1).*

Every model has at least one weakly connected component, and every finite model has a finite number of weakly connected components. Moreover, a model is weakly connected, precisely if the number of weakly connected components is 1.

The following lemma will facilitate the proof of the main theorem of this section.

Lemma 1 *The following hold.*

1. $\mathcal{M} \models (A \perp\!\!\!\perp B|C)$ *iff for all finite, acyclic* $\mathcal{N} \subseteq \mathcal{M}$, $\mathcal{N} \models (A \perp\!\!\!\perp B|C)$
2. $\mathcal{M} \models (A = B)$ *iff for all finite, acyclic* $\mathcal{N} \subseteq \mathcal{M}$, $\mathcal{N} \models (A = B)$
3. *For all weakly connected* $\mathcal{N} \subseteq \oplus\mathfrak{C}$ *there is an* $\mathcal{M} \in \mathfrak{C}$ *such that* $\mathcal{N} \subseteq \mathcal{M}$
4. $\oplus\mathfrak{C} \models (A \perp\!\!\!\perp B|C)$ *iff for all* $\mathcal{M} \in \mathfrak{C}$, $\mathcal{M} \models (A \perp\!\!\!\perp B|C)$.
5. $\oplus\mathfrak{C} \models (A = B)$ *iff for all* $\mathcal{M} \in \mathfrak{C}$, $\mathcal{M} \models (A = B)$.
6. *If* $\mathcal{M}_1, \ldots, \mathcal{M}_n$ *are finite and acyclic, then* $\oplus\{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$ *is finite and acyclic.*

Proof:

1. [$\Rightarrow$]. Suppose that $\mathcal{N} \not\models (A \perp\!\!\!\perp B|C)$ for some $\mathcal{N} \subseteq \mathcal{M}$. Then $\mathcal{N}$ contains a path from a node in $A^{\mathcal{N}}$ to a node in $B^{\mathcal{N}}$, that is not blocked by $C^{\mathcal{N}}$. As $\mathcal{N} \subseteq \mathcal{M}$, the same unblocked path is also present in $\mathcal{M}$, hence $\mathcal{M} \not\models (A \perp\!\!\!\perp B|C)$.

   [$\Leftarrow$] Suppose $\mathcal{M} \not\models (A \perp\!\!\!\perp B|C)$. Then there must be an acyclic path from a node in $A^{\mathcal{M}}$ to a node in $B^{\mathcal{M}}$ that is not blocked by $C^{\mathcal{M}}$ (if there is a cyclic path, then there must also be an acyclic path). Pick such an acyclic path, say $h_1, \ldots, h_n$. For each, h_i, if h_i is a head-to-head point on the graph, then there must be a directed path from h_i to a node in $C^{\mathcal{M}}$. For each head-to-head point h_i pick such a path. Now define $\mathcal{N}$ to be the submodel of $\mathcal{M}$ containing $h_1, \ldots, h_n$ plus the nodes on the chosen paths from head-to-head points h_i to nodes in $C^{\mathcal{M}}$ (and all the arrows between them). As shown by Geiger and Pearl (1990), $\mathcal{N} \not\models (A \perp\!\!\!\perp B|C)$. Furthermore, $\mathcal{N}$ is finite, acyclic and weakly connected.

2. [$\Rightarrow$]. Suppose $\mathcal{N} \not\models (A = B)$ for some $\mathcal{N} \subseteq \mathcal{M}$. Then there must be a v that is in $A^{\mathcal{N}}$ but not in $B^{\mathcal{N}}$, or vice versa. Since $\mathcal{N} \subseteq \mathcal{M}$ (and $v \in \mathcal{N}$), we know that $v \in A^{\mathcal{M}}$ iff $v \in A^{\mathcal{N}}$ (and likewise for B). From this, it follows that $\mathcal{M} \not\models (A = B)$.

   [$\Leftarrow$] Suppose $\mathcal{M} \not\models (A = B)$. Then there must be an v such that v is in $A^{\mathcal{M}}$ but not in $B^{\mathcal{M}}$ or vice versa. Let $\mathcal{N} = (\{v\}, \emptyset, \lambda x.I(x) \cap \{v\})$. Then $\mathcal{N} \subseteq \mathcal{M}$ and $\mathcal{N} \not\models (A = B)$. Trivially, $\mathcal{N}$ is finite, acyclic and weakly connected.

3. By contraposition. Suppose that $\mathcal{N} \subseteq \oplus\mathfrak{C}$ and there is no $\mathcal{M} \in \mathfrak{C}$ such that $\mathcal{N} \subseteq \mathcal{M}$. Then $\mathcal{N}$ must contain nodes originating from at least two models (elements of $\mathfrak{C}$). From this, it follows that $\mathcal{N}$ cannot be weakly connected.

4. From (1) and (3), by that fact that the $N \subseteq \mathcal{M}$ constructed in the proof of (1) is weakly connected.

5. From (2) and (3), by that fact that the $N \subseteq \mathcal{M}$ constructed in the proof of (2) is weakly connected.

6. Finiteness follows from the Pigeon Hole Principle, acyclicity from (3).

□

Theorem 1 *Given a model $\mathcal{M}$ and a finite set of variables P there is a finite, acyclic, weakly connected model $\mathcal{N}$ that agrees with $\mathcal{M}$ on all sentences built from variables in P.*

Proof: Let a model $\mathcal{M}$ and a finite set of variables P be given. Up to equivalence, only finitely many terms can be constructed from the set of variables P. Let $\mathbb{P}$ be a finite (non-empty) set of terms constructed from variables in P, such that any term constructed from variables in P is equivalent to a term in $\mathbb{P}$.

We will construct a finite, acyclic, weakly connected model that is P-equivalent to $\mathcal{M}$. The construction goes in three steps.

Step 1: Collecting witnesses.
Suppose $\mathcal{M} \not\models (A \perp\!\!\!\perp B|C)$ for some $A, B, C \in \mathbb{P}$. Then by Lemma 1, there must be a finite, acyclic model $\mathcal{N} \subseteq \mathcal{M}$ such that $\mathcal{N} \not\models (A \perp\!\!\!\perp B|C)$. Choose one and call it $\mathcal{M}_{A,B,C}$. Likewise, suppose $\mathcal{M} \not\models (A = B)$

for some $A, B \in \mathbb{P}$. Then by Lemma 1, there must be a finite, acyclic model $\mathcal{N} \subseteq \mathcal{M}$ such that $\mathcal{N} \not\models (A = B)$. Choose one and call it $\mathcal{M}_{A,B}$. Let $\mathfrak{C}$ be the collection of all these chosen witnessing submodels, i.e., $\mathfrak{C} = \{\mathcal{M}_{A,B,C} \mid A, B, C \in \mathbb{P} \,\&\, \mathcal{M} \not\models (A \perp\!\!\!\perp B|C)\} \cup \{\mathcal{M}_{A,B} \mid A, B \in \mathcal{P} \,\&\, \mathcal{M} \not\models (A = B)\}$. Since $\mathbb{P}$ is finite and non-empty, it follows that $\mathfrak{C}$ is also finite and non-empty. Furthermore, it follows from Lemma 1 that every model in $\mathfrak{C}$ satisfies every *positive* basic proposition satisfied by $\mathcal{M}$.

Step 2: Taking the disjoint union.
Let $\mathcal{N}$ be the disjoint union $\oplus\mathfrak{C}$ (in the case that $\mathfrak{C} = \emptyset$, let $\mathcal{N}$ be any model with one node and no arrows). It follows from Lemma 1 that $\mathcal{N}$ agrees with $\mathcal{M}$ on all basic sentences containing only terms in $\mathbb{P}$ (and therefore on all basic sentences containing only variables in P). By induction, it follows that $\mathcal{N}$ and $\mathcal{M}$ agree on arbitrary sentences containing only variables in P. Furthermore, $\mathcal{N}$ is finite and acyclic.

Step 3: Making the model weakly connected.
If $\mathcal{N}$ is weakly connected, then we're done. If not, then let n be the number of weakly connected components of $\mathcal{N}$. We know that $n \geq 2$. Pick two nodes w, v from different components, and extend $\mathcal{N}$ by adding two worlds u, t: If $\mathcal{N} = (V, R, I)$, then $\mathcal{N}'$ becomes $(V \cup \{u, t\}, R \cup \{(w, u), (u, t), (v, t)\}, I')$. The new I' is an extension of I such that for all variables X, $u \in I'(X)$ iff $w \in I'(X)$ and $t \in I'(X)$ iff $w \in I'(X)$.

The new model $\mathcal{N}'$ is $\mathcal{LD}$-indistinguishable from $\mathcal{N}$ but the number of weakly connected component has decreased by one. By repeating this procedure, we reduce the number of weakly connected components to 1, in which case the model is weakly connected. □

Theorem 2 *Every satisfiable sentence is satisfied by some finite, acyclic, weakly connected model.*

Proof: Let P be the set of proposition letters occurring in ϕ, and apply the above theorem. □

3.3 Embedding $\mathcal{LD}$ in Converse PDL

There is a LinearTime, satisfiability preserving translation procedure from $\mathcal{LD}$ to *Converse Propositional Dynamic Logic* (cf. Appendix A). Using this translation, model checking and satisfiability checking algorithms, proof systems and complexity results for $cPDL$ can be applied to $\mathcal{LD}$. There are three reasons for choosing a modal logic (and in particular cPDL) as a target language for the translation procedure, instead of e.g. first-order logic.[3]

Table 6: Translation from $\mathcal{LD}$ to $cPDL$

$$
\begin{array}{rcl}
(X)^{\#} & = & X \\
(\emptyset)^{\#} & = & \bot \\
(-A)^{\#} & = & \neg(A^{\#}) \\
(A \cap B)^{\#} & = & A^{\#} \wedge B^{\#} \\
(A \cup B)^{\#} & = & A^{\#} \vee B^{\#} \\
 & & \\
(A = B)^{\#} & = & [o](A^{\#} \leftrightarrow B^{\#}) \\
(A \perp\!\!\!\perp B|C)^{\#} & = & [o](A^{\#} \rightarrow \neg\langle r_{C^*}\rangle B^{\#}) \\
(\neg\phi)^{\#} & = & \neg\phi^{\#} \\
(\phi \wedge \psi)^{\#} & = & \phi^{\#} \wedge \psi^{\#} \\
(\phi \vee \psi)^{\#} & = & \phi^{\#} \vee \psi^{\#} \\
(\phi \rightarrow \psi)^{\#} & = & \phi^{\#} \rightarrow \psi^{\#}
\end{array}
$$

1. Intuitively d-separation is a very graphical notion, talking about *paths in a graph*. Also, Theorem 2 indicates a connection between $\mathcal{LD}$ and modal logics. Indeed, $\mathcal{LD}$ resembles the basic temporal logic K_t in this respect.

2. More importantly, first order logic is not expressive enough for our purposes. D-separation, as it is defined, concerns paths of arbitrary length, and first-order logic does not offer the expressive power to handle this, while $cPDL$ does.[4]

3. $cPDL$ is decidable whereas first-order logic is not (i.e., there is a decision procedure for determining whether a $cPDL$ sentence is valid or not).

It might come as a surprise to the reader that we will translate a term-based language into a propositional language. The statements in the language of $\mathcal{LD}$ are essentially predications over terms. $cPDL$ on the other hand, being a propositional modal logic, consists only of propositions and modalities. In fact, both terms and statements of the original language are translated into propositions (i.e., sets of nodes).

[3] The choice to use a modal logic as target language, was not based on the fact that modal logics have been used in the past to analyse epistemic propositions.

[4] Of course, $cPDL$ is not *more* expressive than first order logic (otherwise, it would be undecidable). Nevertheless, there are things that one can express in $cPDL$, that cannot be expressed in first order logic. Cf. Van Benthem (1984) for a detailed comparison of modal logics and first order logic with respect to their expressiveness.

The actual translation procedure is given in Table 6. The PDL modality r is used to represent the arrows between the nodes in the belief network (i.e., r corresponds to the **down** action of Figure 2). Furthermore, r_ϕ is used as an abbreviation for $?\neg\phi;((?\top \cup ((r;?\neg\phi)^*;r;?\langle r^*\rangle\phi));\bar{r};?\neg\phi)^*;(r;?\neg\phi)^*$. This formula is in fact the regular expression corresponding to the *d-separation automaton* depicted in Figure 2 (where ϕ plays the role of Z).[5]. The rest of the translation procedure is relatively straightforward. Two primitive modalities are used: r and o. In the PDL models, r will be interpreted as the actual accessability relation (i.e., the edges in the graph), whereas o will be interpreted as the universal modality, relating every node to every other node.

Theorem 3 *For every model $\mathcal{M}$ and node w, the following are equivalent (where $[o]$ is interpreted as the universal modality).*

1. $\mathcal{M} \models_{\mathcal{LD}} \phi$
2. $\mathcal{M}, w \models_{cPDL} \phi^{\#}$.

Proof: (sketch)

Step one: terms
The first step is to prove that for any term A and any node w, $\mathcal{M}, w \models_{cPDL} A^{\#}$ iff $w \in A^{\mathcal{M}}$. The proof is completely straightforward, by induction on A.

Step two: atomic statements
The main part of the theorem consists of the fact that $\mathcal{M}, w \models_{cPDL} \langle r_\phi\rangle\psi$ iff there is a path from w to some other node v such that the path is not blocked by $[[\phi]]^{\mathcal{M}}$ and $\mathcal{M}, v \models \psi$ (where $[[\phi]]^{\mathcal{M}}$ is the extension of ϕ in the model $\mathcal{M}$). This follows directly from the fact that r_ϕ is the regular expression corresponding to the d-separation automaton for $[[\phi]]^{\mathcal{M}}$.

Step three: arbitrary statements
Finally, it will be proven that for any $\mathcal{LD}$ formula ϕ, $\mathcal{M} \models_{\mathcal{LD}} \phi$ iff $\mathcal{M}, w \models_{cPDL} \phi^{\#}$. This is again straightforward, by induction (the atomic cases where ϕ the form $(A \perp\!\!\!\perp B|C)$ and $(A = B)$ follows directly from Step 1 and 2). □

As an immediate corrolary, we have that model checking for $\mathcal{LD}$ is LinearTime, i.e., $O(|M||\phi|)$ (cf. also Geiger, Verma and Pearl, 1990). This follows from the fact that both our translation procedure and model checking for $cPDL$ are LinearTime.

[5]For more information on the relation between regular expressions and finite state automata and the translation from one to the other, the reader is referred to any introductory textbook on automata theory, eg. Linz (1996).

Corrolary 1 *$\mathcal{LD}$ model checking is LinearTime.*

We can also use the translation for deciding the validity of $\mathcal{LD}$ sentences. There are ExpTime decision procedures for validity of converse PDL on the class of all frames. Now, in combination with our result from the previous section, this decision procedure can also be applied to $\mathcal{LD}$. In order to find out whether a sentence is satisfiable, simply translate it to $cPDL$, replace all occurrences of the modality o by $(r \cup \bar{r})^*$ and test the satisfiability of the resulting formula.[6] All in all, this can be performed in ExpTime.

Corrolary 2 *$\mathcal{LD}$ validity can be decided in ExpTime.*

Also, there is a complete Hilbert-style axiomatisation of validity in cPDL (Harel, 1984), as well as a complete tableaux calculus (De Giacomo and Massacci, 2000). Both of these can be used to find and represent proofs of validity of (translations of) $\mathcal{LD}$ sentences.

4 $\mathcal{LD}^+$: extending the language with singleton variables

In this section, we will extend the language with special kinds of variables, that can refer only to singleton sets. There are two reasons for extending the language in this way.

1. Some properties of d-separation that have been discussed in the literature, can only be expressed using singleton-variables (cf. Table 3).
2. The following question has been asked by Pearl (1988). Given a finite set of random variables W and a set of triples $I \subseteq \wp(W) \times \wp(W) \times \wp(W)$, what are the neccessary and sufficient requirements on I such that I is the extension of the d-separation relation in some finite directed acyclic graph.

   If we extend the language of $\mathcal{LD}$ with singleton variables, then the question whether a given set of triples I is the extension of the d-separation relation in some directed acyclic graph, becomes an instance of the satisfiability problem.[7]

The language of $\mathcal{LD}^+$ is that of $\mathcal{LD}$, extended with singleton variables $\alpha, \beta, \gamma, \ldots$. Semantically, these singleton variables are interpreted in the same way as

[6]On weakly connected frames, o is equivalent to $(r \cup \bar{r})^*$. Since $\mathcal{LD}$ has the weakly connected model property, we can simply replace o with $(r \cup \bar{r})^*$.

[7]Using singleton variables, every such set of triples I can be completely described in one sentence.

normal (set) variables, except that we require of the models that they assign singeton sets to the singleton variables.[8]

Although this addition might seem quite innocent, it will make things more complicated. In particular, the finite, weakly-connected model property no longer holds, because the addition of singleton variables makes it possible to distinguish between cyclic and acyclic models. Also, weak connectedness can no longer be assumed. However, we still have the finite model property.

Theorem 4 *$\mathcal{LD}^+$ has the finite model property*

Proof: The appropriate construction consists of the first two steps of the proof of Theorem 1, where the operation of *disjoint union* is replaced by *ordinary union*.[9] □

Since the only essential difference between $\mathcal{LD}$ and $\mathcal{LD}^+$ is the class of models that is considered, the results of the previous section concerning model checking complexity and algorithms, directly transfer.[10]

Corrolary 3 *LD^+ model checking is LinearTime.*

The other results don't immediately transfer. Since $cPDL$ has no correspondent of singleton variables, we need to choose a richer language as the target language for our translation procedure. Moreover, since $\mathcal{LD}^+$ lacks the acyclic model property, and we want to generate all validities on the class of finite *acyclic* models, we need to be able to express acyclicity somehow in the target language. Finally, we also need the universal modality in our target language, since we can no longer assume weak connectedness. There are two candidate target languages: *Converse PDL extended with nominals, intersection and universal modality*, and the *Hybrid μ-calculus*. I will discuss both.

[8]Singleton variables have been studied extensively in a particular branch of modal logic called *Hybrid Logic*, where they are referred to as *nominals*. Blackburn (2000) provides a good introduction to hybrid logics.

[9]There are two reasons for using ordinary union instead of disjoint union. First, in the proof of theorem 2 we wanted to end up with an acyclic model. By taking the disjoint union of a collection of acyclic models, acyclicity of the resulting model was achieved. However, in the case of theorem 4 acyclicity is not required. Second, we now have to consider singleton variables. In order to guarantee that every singleton variable is assigned a singleton set, we need to use ordinary union, not disjoint union.

[10]For the model checking task, a model is given beforehand, together with a formula. We simply assume that the given model assigns singleton sets to singleton variables.

First let us consider the extension of $cPDL$ with nominals, intersection and universal modality discussed by Passy and Tinchev (1991). Nominals are a special kind of propositions that are true at precisely one point in the model. As the reader will realise, they are just the PDL counterpart of the singleton variables that we have in $\mathcal{LD}^+$. Thus, it is straightforward to extend the translation procedure by translating singleton variables into nominals. As for the acyclicity condition, this can be expressed in terms of intersection. Let $CYCLE = \langle (r; r^*) \cap ?\top \rangle \top$. Then $CYCLE$ holds at a state just in case it lies on a cycle. Now, we can express acyclicity of the graph by $[o]\neg CYCLE$ (where o is the universal modality).

Unfortunately, this extension of $cPDL$ is undecidable (cf. Blackburn, De Rijke and Venema (2001)). Still, Passy and Tinchev (1991) provide a complete Hilbert-style axiomatisation. We can at least use this axiomatisation to find and represent proofs of the validity of (translations of) $\mathcal{LD}^+$ sentences.

The other candidate target language is the *Hybrid μ-calculus* (Sattler and Vardi, 2001). This language is again an extension of $cPDL$, it also has nominals and universal modality, but instead of intersection, it has *fixed point operators* that can be used to express acyclicity. The formula that does the trick is $[o](\mu x.[r]x)$.[11] The results in the literature for the hybrid μ-calculus are in a sense opposite to those for $cPDL$ with nominals and intersection: there is *no* complete axiomatisation at hand, but there *is* an ExpTime decision procedure for checking validity, based on automata-theoretic techniques (Sattler and Vardi, 2001). This gives us the following result.

Corrolary 4 *$\mathcal{LD}^+$ validity can be decided in ExpTime.*

5 Discussion

The central theme of this paper is a translation procedure from d-separation statements to formulas of (extensions of) Propositional Dynamic Logic. Immediate corrolaries of this result concern complexity issues. However, the results presented in this paper have other interesting applications.

An important theoretical question that is still open concerns the axiomatic characterization of the d-separation. D-separation is intended to approximate the notion of conditional independence in probability

[11]In fact, this formula expresses *well-foundedness*. However, on finite models, well-foundedness and acyclicity are equivalent, by the Pigeon Hole Principle.

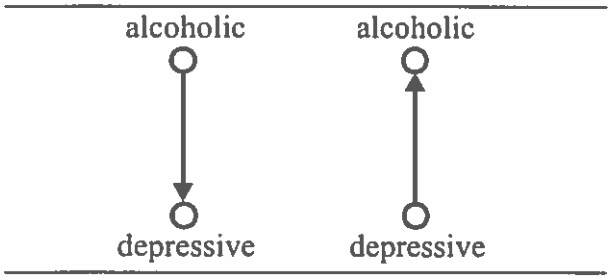


Figure 3: Two indistinguishable models

distributions as closely as possible. However, it turns out that in order for d-separation to correspond exactly to conditional independence, some additional assumptions must hold. The question is: *what are these assumptions precisely?* Studeny (1992) has shown that conditional independence has no finite complete axiomatization. However, this doesn't preclude complete axiomatizations of d-separation. The present paper provides further means to find such an axiomatisation.

Another interesting question that presents itself through the connection between d-separation and modal logic, is: *when are two models distinguishable?* For example, there is no formula in the language of $\mathcal{LD}$ (or $\mathcal{LD}^+$) that can distinguish the two models depicted in Figure 3. For modal logics, distinguishability of models has been characterized in terms of *bisimulation* (Van Benthem, 1983). It would be very interesting to see if a similar notion exists for d-separation. On a more philosophical level, this relates to *the limits of empirical research*: no matter how many correlations we observe, we will never be able to distinguish certain causal models.[12]

As for practical applications of the results presented in this paper, it is potentially useful to look at tools and algorithms that have been developed in the *automatic software verification* community for reasoning about PDL and related languages. To give an example, the use of *binary decision diagrams* has turned out to be very helpful in the implementation of efficient model checkers for dynamic logics, cf. Huth and Ryan (2000). Perhaps tools like binary decision diagrams have useful applications in the world of belief networks as well.

[12] In general, such models *can* be distinguished by performing experiments, i.e., by directly manipulating the random variables involved. However, in the case of Figure 3, no appropriate experiments can resolve the issue. Cf. Pearl (2000) for a discussion on how to model experiments within the framework of belief networks.

Acknowledgements

I wish to thank Patrick Blackburn, Paul Dekker, Jan van Eijck, Darrin Hindsill and Maarten Marx for their useful comments and suggestions.

References

Johan van Benthem (1983). *Modal logic and classical logic.* Milano: Bibliopolis.

Johan van Benthem (1984). Correspondence theory. In D. Gabbay and F. Guenthner (Eds.), *Handbook of Philosophical Logic*, Volume II. Reidel.

Patrick Blackburn (2000). Representation, Reasoning, and Relational Structures: a Hybrid Logic Manifesto. *Logic Journal of the IGPL 8*(3), 339–625.

Patrick Blackburn, Maarten de Rijke and Yde Venema (2001). *Modal logic.* Cambridge, UK: Cambridge University Press.

Dan Geiger and Judea Pearl (1990). On the logic of causal models. In R. Shachter, T. Levitt, L. Kanal and J. Lemmer (Eds.), *Uncertainty in Artificial Intelligence 4*, New York, NY, pp. 3–14. Elsevier.

Dan Geiger, Thomas Verma and Judea Pearl (1990). D-separation: from theorems to algorithms. In R. Shachter, L. Kanal and J. Lemmer (Eds.), *Uncertainty in Artificial Intelligence 5*, New York, NY, pp. 139–148. Elsevier.

Guiseppe de Giacomo and Fabio Massacci (2000). Combining deduction and model checking into tableaux and algorithms for Converse PDL. *Information and computation 162*(102), 117–137.

David Harel (1984). Dynamic Logic. In D. Gabbay and F. Guenthner (Eds.), *Handbook of Philosophical Logic.* Dordrecht: Reidel.

Michael Huth and Mark Ryan (2000). *Logic in Computer Science: Modelling and Reasoning about Systems.* Cambridge, England: Cambridge University Press.

Peter Linz (1996). *An Introduction to Formal Languages and Automata* (Second edition ed.). Lexington, Massachusetts: D.C.Heath and Company.

Solomon Passy and Tinko Tinchev (1991). An essay in combinatory dynamic logic. *Information and computation 93*, 263–332.

Judea Pearl (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference.* San Mateo, CA: Morgan Kaufmann.

Table 7: Semantics of Converse PDL

$\mathcal{M}, w \models p$	iff	$w \in I(p)$
$\mathcal{M}, w \models \neg\phi$	iff	$\mathcal{M}, w \not\models \phi$
$\mathcal{M}, w \models \phi \wedge \psi$	iff	$\mathcal{M}, w \models \phi$ and $\mathcal{M}, w \models \psi$
$\mathcal{M}, w \models \langle\alpha\rangle\phi$	iff	there is a $u \in W$ such that $w[\alpha]^{\mathcal{M}} u$ and $\mathcal{M}, u \models \phi$
$w[r]^{\mathcal{M}} u$	iff	$(w,u) \in R_r$
$w[\bar{r}]^{\mathcal{M}} u$	iff	$(u,w) \in R_r$
$w[?\phi]^{\mathcal{M}} u$	iff	$\mathcal{M}, w \models \phi$ and $u = w$
$w[\alpha \cup \beta]^{\mathcal{M}} u$	iff	$w[\alpha]^{\mathcal{M}} u$ or $w[\beta]^{\mathcal{M}} u$
$w[\alpha;\beta]^{\mathcal{M}} u$	iff	there is a $v \in W$ such that $w[\alpha]^{\mathcal{M}} v$ and $v[\beta]^{\mathcal{M}} u$
$w[\alpha^*]^{\mathcal{M}} u$	iff	$w[\alpha]^{\mathcal{M}} \cdots [\alpha]^{\mathcal{M}} u$

Judea Pearl (2000). *Causality : Models, Reasoning, and Inference.* Cambridge University Press: Cambridge University Press.

Ulrike Sattler and Moshe Vardi (2001). The hybrid mu-calculus. In R. Goré, A. Leitsch and T. Nipkow (Eds.), *Proceedings of IJCAR'01*, Seattle, WA. Springer Verlag.

Milan Studeny (1992). Conditional independence relations have no finite complete characterization. In S. Kubik and J. Visek (Eds.), *Transactions of the 11th Prague Conference vol. B*, pp. 377–396. Kluwer.

Appendix A: Converse PDL

The language of Converse PDL is built up from proposition letters $p, q, r, \ldots$ and modalities $r, s, t, \ldots$ in the following way.[13]

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \psi \mid \langle\alpha\rangle\phi$$
$$\alpha ::= r \mid \bar{r} \mid ?\phi \mid \alpha \cup \beta \mid \alpha;\beta \mid \alpha^*$$

Sentences of this language are interpreted with respect to multi-modal Kripke structures of the form (W, R, I), where R is a function that assigns to each basic modality an accessability relation, and I is a valuation function that assigns a set of worlds to each proposition letter. With respect to such a model $\mathcal{M} = \langle W, R, I\rangle$, the sentences of the language are interpreted as in Table 7.

Converse PDL has many nice properties. For instance, model checking can be performed in LinearTime (i.e., given a finite model $\mathcal{M}$ and a sentence ϕ, it can be decided in $O(|\mathcal{M}||\phi|)$ whether the sentences is true or false in the model). Satisfiability can be decided in ExpTime. There is a complete Hilbert style axiomatisation (Harel, 1984) and a complete tableaux calculus (De Giacomo and Massacci, 2000).

Various extension of the language have been proposed. In particular, Passy and Tinchev (1991) discuss an extension of Converse PDL that is relevant for present purposes, using nominals, universal modality and intersection. The nominals $i, j, k, \ldots$ are special kinds of propositions, that are true at precisely one point in the model.[14] Intersection has the obvious semantics: $\mathcal{M}, (w,v) \models \alpha \cap \beta$ iff $\mathcal{M}, (w,v) \models \alpha$ and $\mathcal{M}, (w,v) \models \beta$. Finally, the universal modality o is The usual abbreviations are used: $[\alpha]\phi \equiv \neg\langle\alpha\rangle\neg\phi$, $\bot \equiv p \wedge \neg p$, etc. added to the set of basic modalities. It is interpreted as the universal relation between worlds. I.e., $\langle o\rangle\phi$ is true at a point w if there is another point v in the model, such that ϕ is true at v.

For this extension of $cPDL$ there is a complete axiomatization, cf. Passy and Tinchev (1991). However the logic is in general no longer decidable (an undecidability proof can be found in Blackburn, De Rijke and Venema (2001)).

Another relevant extension of Converse PDL that has been proposed, is the Hybrid μ-calculus, which is obtained by adding nominals, universal modality and a *least fixed point* operator μ (Sattler and Vardi, 2001). A sentence of the form $\mu x.\phi(x)$ is interpreted as the strongest proposition (smallest set of worlds) ψ such that $\psi \leftrightarrow \phi(\psi)$ is globally true on the model.

For example, the sentence $\mu x.p \wedge [r]x$ is interpreted as the strongest proposition ψ such that $\psi \leftrightarrow p \wedge [r]\psi$ is globally true on the model. As the reader can check by unfolding this fixed point equation, ψ corresponds precisely to the formula $[r^*]p$.

The results in the literature for the hybrid μ-calculus are in a sense opposite to those for $cPDL$ with nominals and intersection: there is *no* complete axiomatisation at hand, but there *is* an ExpTime decision procedure for checking validity, based on automata-theoretic techniques (Sattler and Vardi, 2001).

[13] Notice that, in this definition, α is used as a meta-variables over modalities, and not as a singleton variable.

[14] Note that this is just a requirement on the models that are considered. In the definition of the satisfaction relation, the nominals behave just like ordinary propositions.

Solving QBF with SMV

Francesco M. Donini
Dipartimento di Elettrotecnica ed Elettronica
Politecnico di Bari
Via Re David 200
Bari, Italia

Paolo Liberatore
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113
Roma, Italia

Fabio Massacci
Dipartimento di Ingegneria Civile e Ambientale
Università di Trento
Via Sommarive, 14
Povo (Trento), Italia

Marco Schaerf
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113
Roma, Italia

Abstract

The possibility of solving the Quantified Boolean Formulae (QBF) problems using the SMV system is a consequence of two well-known theoretical results: the membership of QBF to PSPACE, and the PSPACE-hardness of LTL (and therefore, of SMV). Nevertheless, such results do not imply the existence of a reduction that is also of practical utility. In this paper, we show a reduction from QBF to SMV that is linear (instead of cubic), and uses a constant-size specification.

This new reduction has three applications the previous one has not: first, it allows for solving QBF problems using SMV-like systems, which are now more developed than direct QBF solvers; second, we can use it to verify whether the performance behavior of direct QBF solvers is intrinsic of the problem, or rather an effect of the solving algorithm; third, random hard SMV instances can be easily generated by reduction from QBF hard instances (whose generation method is now established).

1 Introduction

In recent years a number of solvers for Quantified Boolean Formulae (QBFs) have been proposed in the literature [CSGG00, Rin99, EETW00, FMS00]. QBFs are a natural extension of satisfiability problems in classical propositional logic (SAT), and are of great theoretical interest since they allow to precisely characterize all computational complexity classes in the polynomial hierarchy and PSPACE [Sto76]. From a knowledge representation point of view, it is well known that most problems in propositional reasoning, such as planning, nonmonotonic reasoning, abduction, and diagnosis can be reduced to the problem of checking the truth of a quantified boolean formula.

An obviously interesting problem is to find out whether the phenomena of the existence of a steep state transition and of the easy-hard-easy pattern, which are found in SAT problems, also arise in QBF, and if they are similarly related. The initial investigations based on the Evaluate algorithm [CSGG00] have shown that both phenomena exist in QBFs, even though not as evident as in SAT. Further investigations by Gent and Walsh [GW99] and others have confirmed this initial investigation.

In this paper we show a very simple and efficient reduction from the problem of checking the truth of a QBF into model-checking of an LTL (linear temporal logic) or a CTL (computation tree logic) formula over an SMV model. In the following section we briefly present LTL, CTL, and SMV. The reader interested in a more formal and detailed presentation should refer to McMillan's thesis [McM93] and Huth and Ryan's book [HR00] that contain a detailed exposition. Implementing a translator based on this reduction allows for using the SMV system as a QBF solver. A first benefit is that, since the SMV system is widely used, well engineered, and efficient, this method may be more efficient than direct QBF solving.

A second benefit is the effect of SMV being based on a completely different technology than most of the current QBF solvers, which are usually variations of the DLL procedure [DLL62], and are therefore way too similar to prove that some computational properties (such as the easy-hard-easy pattern) are intrinsic of the problem.

A third important benefit of the proposed reduction is for the benchmark generation and system evaluation in model checking. Indeed, currently one is forced to choose between two alternatives:

1. generate computationally challenging benchmarks using the problems constructed from the theoretical results; these problems (model-checking explicit Kripke structures) are not interesting for evaluating actual systems, which take a different input (an SMV specification);
2. use the large number of "real world" problems (written using SMV specifications); unfortunately, there is no way to know whether these problems are really representative of a broader class of problems and really computationally challenging.

We would like to have the best of both approaches: *generation of computationally challenging (and controllable) benchmark with model-checking specifications actually used in practice.* The reduction we propose allows for generating benchmarks from random QBF instances, which can be generated in a way that guarantees their computational properties. Namely, the critical parameters for the generation of hard instances of QBF have been deeply studied [CSGG00].

From a theoretical point of view, this reduction implies that CTL and LTL model checking Kripke specifications written in SMV is PSPACE-hard *even when the specification has constant size* (known reductions employ specifications of polynomial size). However, the benefit of this reduction is more practical than theoretical: besides having a new method for solving QBF instances, we can now generate model checking problems whose hardness can be somehow "controlled". This allows for testing the empirical effectiveness of various model checking optimizations. QBFs offer complete problems for all classes in the polynomial hierarchy [Sto76], and therefore very suitable for the experimental analysis, as they include problems that belong to a wide range of complexity classes [CSGG00, EETW00], a property not easily found with direct experimental analysis. See for instance the difficulties noted by Daniele, Giunchiglia, and Vardi [DGV99] for the experimental comparison of LTL automata generation algorithms.

```
MODULE main
VAR
  request : boolean;
  state   : {ready, busy};
ASSIGN
  init(state) := ready;
  next(state) := case
        state = ready & request = 1 : busy;
                                    : {ready, busy};
        esac;

SPEC
  AG(request -> AF state = busy)
```

Figure 1: An example SMV program

We remark that a proof of PSPACE completeness of LTL, implying the existence of a polynomial reduction from QBF to SMV, already exists [SC85]. However, this reduction requires a cubic encoding and requires the usage of the *until* modality, while our reduction is linear and employs the temporal modalities *globally* and *next* only. This is why the new reduction is more suited to a practical use.

2 Preliminaries

In this section we briefly introduce the model-checking system SMV; the following presentation is adapted from the NuSMV manual [CCO+]. Further details on SMV are in McMillan's book [McM93], while NuSMV is described in the paper by Cimatti et al. [CCGR00].

The primary purpose of an SMV specification is to describe the transition relation of a Kripke structure. Any expression in the propositional calculus can be used to describe this relation. This provides a great deal of flexibility, and at the same time a certain danger of inconsistency. For example, the presence of a logical contradiction can result in a deadlock – a state or states with no successor. This can make some specifications vacuously true, and makes the description unimplementable. While the model checking process can be used to check for deadlocks, it is best to avoid the problem when possible by using a restricted description style. The SMV system supports this by providing a parallel-assignment syntax. The semantics of assignment in SMV is similar to that of single assignment data flow language. By checking programs for multiple parallel assignments to the same variable, circular assignments, and type errors, the interpreter insures that a program using only the assignment mechanism is implementable. Consequently, this fragment of the language can be viewed as a description language, or a programming language.

Consider the SMV program in Figure 1. The first part defines the Kripke structure. The space of states of the Kripke structure is determined by the declarations of the state variables (in the above example `request` and `state`). The variable `request` is declared to be of (predefined) type boolean. This means that it can assume the (integer) values 0 and 1. The variable `state` is a scalar variable, which can take the symbolic values `ready` or `busy`.

The following assignment sets the initial value of the variable state to ready. The initial value of request is completely unspecified, i.e. it can be either 0 or 1.

The transition relation of the Kripke structure is expressed by defining the value of variables in the next state (i.e. after each transition), given the value of variables in the current states (i.e. before the transition). The `case` segment sets the next value of the variable `state` to the value `busy` (after the column) if its current value is `ready` and `request` is 1 (i.e. true). Otherwise (the 1 before the column) the next value for `state` can be any in the set `{ready,busy}`.

The variable `request` is not assigned. This means that there are no constraints on its values, and thus it can assume any value, which is thus an unconstrained input to the system.

Specifications can be expressed in CTL (Computation Tree Logic). The NuSMV system also allows for specifications in LTL (Linear Temporal Logic). These logics allow a rich class of temporal properties, including safety, liveness, fairness and deadlock freedom, to be specified in concise a syntax.

The keyword SPEC is followed by a CTL formula, that is intended to be checked for truth in the Kripke structure defined above. The intuitive reading of the formula is that every time `request` is true, then in all possible future evolution, eventually `state` must become `busy`.

The two (modal) propositional temporal logics LTL and CTL are used to express temporal properties of the modeled system. LTL is a temporal logic whose underlying model of time is linear. More precisely, every model of LTL is a Kripke structure where all worlds are connected in a (possibly infinite) chain. The syntax of LTL includes three unary modal operators X, F and G and a binary one U. Where the intended meanings are:

- $X\alpha$ means: α is true in the next state;
- $F\alpha$ means: α is true in some future state;
- $G\alpha$ means: α is true in all future states;
- $\alpha U \beta$ means: α is true Until β becomes true.

When a linear model of time is not adequate we can resort to CTL whose semantics is based on branching time. The semantics is given via unrestricted Kripke models. In the contest of CTL a sequence of connected states is called a path. In CTL it is possible to quantify either existentially or universally on the paths.

3 Intuition

In this section we provide some intuitions of the reductions presented in Sections 4.

The truth of the QBF formula F is transformed into a model checking problem for SMV in which the model is made up by a number of (slightly modified) counters, a boolean formula in CNF and a number of modules that non-deterministically set a boolean variable.

In a nutshell, the SMV system works as follows:

- all possible assignments to the universal variables are generated using one binary counter whose digits are the assignments to the universal variables, while the assignments to the existential ones are "guessed" non-deterministically. In each state of the system the satisfaction of the CNF formula is checked. Whenever the CNF formula is not satisfied the counting is halted (no carry is propagated to the next variable).
- The LTL (or CTL) specification simply says that for any execution there exists a state that makes the formula false.

4 From QBF to SMV structures

QBFs extend propositional logic by allowing propositional variables to be quantified over, either existentially or universally. The *evaluation problem* for a QBF is to decide whether a given QBF is true or not.

Let F be the Quantified Boolean Formula over n propositional variables:

$$Qx_1 Qx_2 \ldots Qx_n.(\gamma_1 \wedge \cdots \wedge \gamma_m)$$

where Q is a quantifier (either $\forall$ or $\exists$) and each γ_j is a clause over the variables $\{x_1, \ldots, x_n\}$. The sequence of quantifiers $Qx_1 Qx_2 \ldots Qx_n$ is called the prefix of F and will be denoted as P, while un-quantified boolean formula $\gamma_1 \wedge \cdots \wedge \gamma_m$ is called the matrix of F and will be denoted as E. A clause γ_j with k literals is represented as $l_{j_1} \vee \ldots \vee l_{j_k}$, where each literal l is either a variable or its negation.

```
MODULE forall(carry-in)
VAR
   value : boolean;
ASSIGN
   init (value) := 0;
   next (value) := value ^ carry-in;
DEFINE
   carry-out := value & carry-in;
```

Figure 2: The `forall` module in SMV syntax

```
MODULE exists(carry-in)
VAR
   value : boolean;
ASSIGN
   init (value) := {0,1};
   next (value) :=
     case
        carry-in : {0,1};
        1        : value;
     esac;
DEFINE carry-out := carry-in;
```

Figure 3: The `exists` module in SMV syntax

In the above formulation we do not pose any restriction on the alternation of the quantifiers. For any given variable x_i it is only necessary to know its position (i) in the prefix and whether it is existentially or universally quantified.

We now briefly present our encoding of QBFs in SMV. As usual $\{0,1\}$ means the non-deterministic choice between 0 and 1. Each universally quantified variable is coded in a module `forall(carry-in)`, described in Fig. 2, which is a binary digit of a binary counter with carry [HR00, p.184]. The bit is stored in a variable *value*. The presence of a module `forall(`x_{i+1}`.carry-out)` in the main module corresponds to a universally quantified variable x_i.

The module `exists(carry-in)` in Fig. 3 is used for existentially quantified variables. Its single boolean variable `value` is non-deterministically set to 0 or 1. Whenever a carry (the parameter `carry-in`) changes the value of the inner variable, its value is non-deterministically chosen again. The module never changes the carry, but simply forwards the carry it receives from the inner variable to the outer variable.

The boolean formula is coded in a purely reactive module `formula` (Fig. 4) that evaluates the truth value of matrix.

To build the whole SMV code we simply need to:

1. incorporate the definition of the modules `forall` and `exists`;

```
MODULE formula(v1,.......,vn)

DEFINE
        c1 := "first clause of F";
        c2 := "second clause of F";
        .
        .
        cm := "last clause of F";
        sat := c1 & c2 & ....... & cm;
```

Figure 4: The `formula` Module in SMV syntax

```
MODULE main

VAR
        xn    : QUANT(clauses.sat);
        xn-1  : QUANT(xn.carry-out);
        xn-2  : QUANT(xn-1.carry-out);
        .
        x2    : QUANT(x3.carry-out);
        x1    : QUANT(x2.carry-out);
        clauses : formula(x1.value,...,xn.value);
SPEC AF (!clauses.sat)
```

Figure 5: The `main` Module in SMV syntax

2. instantiate the module `formula` with the clauses in the matrix;

3. in the `main` module (see Fig. 5):

   (a) declare the variables $x_1, \ldots, x_n$ using the appropriate module (either `forall` or `exists`) and a parameter, which is the carry-out of the next module, but for x_n whose parameter is *sat* of the formula module;

   (b) declare the variable *clauses* with the module `formula`, and all the values of $x_1, \ldots, x_n$ as parameters;

   (c) declare the specification.

The specification is independent of the QBF and can be expressed either in LTL or in CTL. In LTL it is $\mathbf{F}(\neg \mathit{clauses.sat})$, while in CTL it is $\mathbf{AF}(\neg \mathit{clauses.sat})$. Either formula is interpreted as: for all execution paths of the system, eventually the matrix will not be satisfied. Then clearly the QBF F is false.

An Example of the Translation

In order to make the translation easier to understand, we show it in action: we convert the simple QBF formula $\forall x_1 \exists x_2 . x_1 \equiv x_2$ into SMV code, where $x_1 \equiv x_2$ is an abbreviation for $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$.

First of all, we have the modules `forall` and `exists`, which are not repeated here because they do not de-

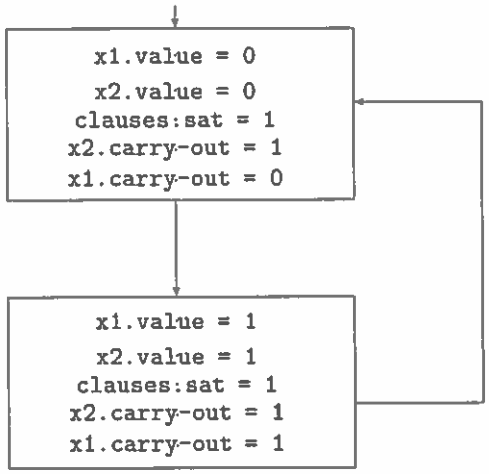


Figure 6: A path that makes the specification false in the translation of $\exists x_1 \forall x_2 . x_1 \equiv x_2$.

pend on the specific formula. The `formula` module is:

```
MODULE formula(x1, x2)
DEFINE
  c1 := x1 | !x2;
  c2 := !x1 | x2;
  sat := c1 & c2;
```

To complete the example, only the `main` module is missing. Here, we specify how the variables are quantified. In particular, each variable is defined to be the result of the appropriate module (`forall` or `exists`), using the carry of the next variable as input (exception made for the last variable, whose module input is the truth value of the formula).

```
MODULE main
VAR
  x2 : exists(clauses.sat);
  x1 : forall(x2.carry-out);
```

Finally, the specification is independent from the formula, and is as follows:

```
SPEC AF (!clauses.sat)
```

In Figure 6 we show an infinite path (in fact a cycle) that makes such a specification false. The initial state encodes the assignment 00 to variables x_1, x_2 of the QBF, while its successor state encodes the assignment 11. Together, 00 and 11 form a proof that $\forall x_1 \exists x_2 . x_1 \equiv x_2$ is a valid QBF.

In order to complete the example, we show how the formula $\exists x_1 \forall x_2 . x_1 \equiv x_2$ is translated (this is almost the same formula as above, but the quantification of variables are swapped). The only point of the SMV code that depends on the quantifiers is the `main` module: now x_1 is existentially quantified while x_2 is universally quantified.

```
MODULE main
VAR
  x2 : forall(clauses.sat);
  x1 : exists(x2.carry-out);
```

The rest of the SMV code is the same as the first example of translation. In Figure 7 the Kripke structure resulting from such translation is depicted. Clearly, for every initial state the specification is now satisfied.

Summarizing, changing the quantification of variables amounts to changing the definition of variables in the `main` module, while changing the matrix leads to a different `formula` module.

5 Correctness of the reduction

To ease notation, we let the symbol 1 stand for $\top$, and 0 for $\bot$. Let $P.E$ be a QBF, with n boolean variables. To determine whether the QBF is valid, one can start by peeling off the outermost quantifier: if it's $\exists x_1$, we choose one of the truth values 1 or 0 and substitute for the newly freed occurrence of x_i; if it's $\forall x_1$, substitute both 1 and 0 for the newly freed occurrences of x_1. In short, while evaluating QBFs we are generating a tree, where existential quantifiers increase the depth, and universal quantifiers force branching. If we always reach a leaf where the assignments of boolean values to variables make the matrix E true, the overall formula F is valid.

If we look at the assignments sequentially, an *assignment* is a sequence of bits; hence, we denote an assignment to variables $x_1, \ldots, x_n$ as a string $b_1 b_2 \cdots b_n \in \{0,1\}^n$, *i.e.*, a sequence of n bits, where b_i is assigned to x_i.

We denote UNIVS(P) the sequence of indexes of the universally quantified variables in the prefix P (in the same order) while EXISTS(P) is the sequence of indexes of the existential quantified variables in P. Given two sequences A and B, the *projection* of B along A, denoted as $\pi_B(A)$, is the subsequence of A obtained by selecting only the elements whose index is in B.

The key of the proof is that we establish a correspondence between the sequence of assignments to the boolean variables generated by the quantifiers of a valid QBF and the states of the SMV model. Given

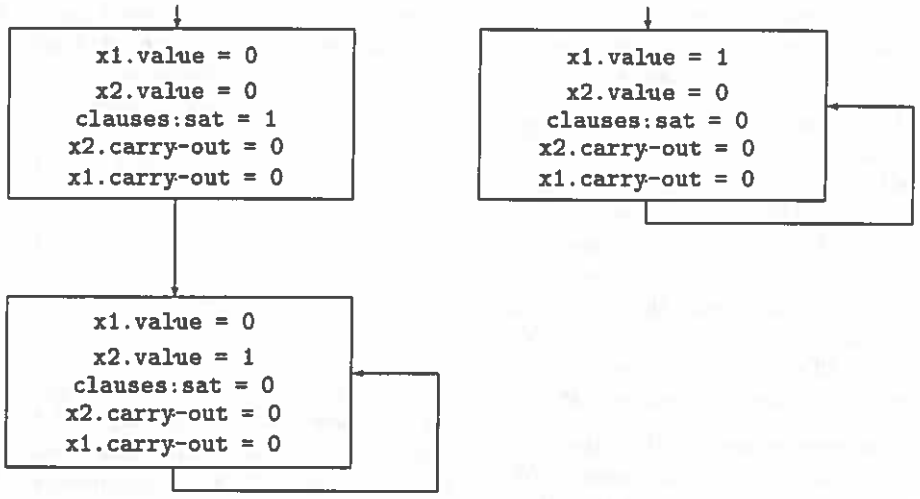


Figure 7: The Kripke structure resulting from the translation of $\forall x_1 \exists x_2 . x_1 \equiv x_2$. The state corresponding to assignment 11 is unreachable from initial states, so it is not represented.

two assignments, we start by defining a relation of successor between them.

Definition 1 (Successor) *Let P be a prefix of a QBF, and let s_1, s_2 be two assignments to variables in P. Then s_2 is a* successor *of s_1 (denoted by $s_1 \rightsquigarrow_P s_2$) if the following conditions hold:*

1. *$\pi_{\text{UNIVS}(P)}(s_2)$ is the successor of $\pi_{\text{UNIVS}(P)}(s_1)$ in the lexicographical order, in $\{0,1\}^{|\text{UNIVS}(P)|}$;*
2. *for any $k \leq n$, if $\pi_{1\cdots k}(\pi_{\text{UNIVS}(P)}(s_1)) = \pi_{1\cdots k}(\pi_{\text{UNIVS}(P)}(s_2))$ then $\pi_{1\cdots k}(\pi_{\text{EXISTS}(P)}(s_1)) = \pi_{1\cdots k}(\pi_{\text{EXISTS}(P)}(s_2))$*

The first condition is equivalent to: the integer numbers defined by taking the value of the universally quantified variables of the sequences must be consecutive. The second condition says that, if the universally quantified variables of two assignments coincide up to an index k, then the existentially quantified variables must coincide up to the same index. In other words, two consecutive assignments $s_1 = a_1 \cdots a_n$ and $s_2 = b_1 \cdots b_n$ can assign a different value to an existentially quantified variable x_i (so, $a_i \neq b_i$) only if there is a universally quantified variable x_j with $j < i$ (*i.e.*, it has a lower index), and $a_j \neq b_j$.

Since the prefix will be clear form the context, in what follows we omit P in $\rightsquigarrow$.

For instance, if $P = \forall x_1 \exists x_2 \forall x_3 \forall x_4 \exists x_5$, then $10010 \rightsquigarrow 10100$. This is because since $\text{UNIVS}(P) = 134$, $\text{EXISTS}(P) = 25$, and $\pi_{\text{UNIVS}(P)}(10010) = 101$, $\pi_{\text{UNIVS}(P)}(10100) = 110$ hence Condition 1 is met. Also, Condition 2 requires that $\pi_{1\cdots 2}(\pi_{\text{UNIVS}(P)}(10010)) = 10 = \pi_{1\cdots 2}(\pi_{\text{UNIVS}(P)}(10100))$ implies that $\pi_{1\cdots 2}(\pi_{\text{EXISTS}(P)}(10010)) = 0 = \pi_{1\cdots 2}(\pi_{\text{EXISTS}(P)}(10100))$, which is indeed true. Instead, $11010 \not\rightsquigarrow 10100$ because Condition 2 is not met.

We now define a sequence of assignments that covers all assignments needed to satisfy a given prefix.

Definition 2 (Consistent sequence) *A sequence of assignments $(s_1, \ldots, s_m)$ is* consistent *with a prefix P if and only if:*

1. *it is made of $m = 2^{|\text{UNIVS}(P)|}$ assignments;*
2. *$\pi_{\text{UNIVS}(P)}(s_1) = 0 \cdots 0$;*
3. *$s_i \rightsquigarrow s_{i+1}$, for every $i = 1, \ldots, n-1$.*

Observe that in a consistent sequence of assignments, the bits assigned to universally quantified variables span from 0 to $2^m - 1$. For the last assignment s_m of the sequence, it must be $\pi_{\text{UNIVS}(P)}(s_m) = 1 \cdots 1$. Not all possible sequences of assignments are consistent with a prefix. For example, let $P = \exists x_1 \forall x_2 \forall x_3$. We have that $\{000, 001, 010, 011\}$ is consistent (we guessed $x_1 = 0$ and tried all possible values of x_2 and x_3) whereas $\{000, 001, 110, 011\}$, and $\{000, 001, 010\}$ are not consistent. Intuitively, once a value is set for an existential variable, all universal variables with higher index must range over all possible truth values

in the successive assignments. In practice, if we have $\exists x_i \forall x_{i+1} \forall x_{i+2} \ldots$ and we set, say, $x_i = 1$ we must use a binary counter to test all possible values of $x_{i+1} x_{i+2}$.

The intuition behind consistent sequences of assignments is that they can represent both a proof of validity of a QBF F, and a path falsifying the specification of the translation of F in SMV. The rest of this section is then devoted to these two issues: on one side, prove that a QBF is valid iff there exists a consistent sequence of assignments all satisfying the matrix; on the other side, prove that the translation of F in SMV has a false specification iff there exists a consistent sequence of assignments all verifying `clauses.sat`.

If the values in an assignment are less than the number of variables n, we call it *partial assignment*. We implicitly assume that the truth values of a partial assignment are used for the first variables, while the last variables are left unassigned.

Definition 3 (Partial Evaluation) *Let E be a boolean formula over variables $x_1, \ldots, x_n$, and let $s = a_1 \cdots a_k$, with $k \leq n$ be a partial assignment. The* partial evaluation *of E according to s, denoted by $E|s$, is the boolean formula obtained by replacing variables $x_1, \ldots, x_k$ with $a_1, \ldots, a_k$ respectively.*

For sake of completeness, we can now recall the inductive definition of validity of a QBF w.r.t. a partial assignment.

Definition 4 *[Validity of a QBF] Let $P.E$ be a QBF, where $P = Q_{k+1} x_{k+1} \cdots Q_n x_n$, and let $s = a_1 \cdots a_k$ be a partial assignment. Formula $P.E|s$ is valid iff one of the following conditions hold:*

1. *If $k = n$ (i.e., P is empty and s is total), then F is valid iff $E|s \equiv 1$;*
2. *If $P = \forall x_{k+1} P'$ (i.e., the first element of P is a universally quantified variable), then F is valid iff both $P'.E|s0$ and $P'.E|s1$ are valid;*
3. *If $P = \exists x_i P'$ (i.e., the first element of P is an existentially quantified variable), then the formula is true iff either $P'.E|s0$ or $P'.E|s1$ is valid.*

Obviously, F is valid iff it is valid w.r.t. the empty assignment.

Theorem 1 *A QBF $P.E$ is valid if and only if there exists a sequence of assignments that is consistent with P, and such that every assignment satisfies E.*

Proof. $\Rightarrow$ Let $P.E$ be valid: then Definition 4 is met. Based on the definition, we recursively construct a sequence of assignments. Let $S(\mathcal{P}, \mathcal{E}, s)$ a function from a (generic) prefix $\mathcal{P}$, a (generic) matrix $\mathcal{E}$, and a partial assignment s, to a sequence of assignments. Let $\circ$ denote the concatenation of sequences of assignments. Then $S(\mathcal{P}, \mathcal{E}, s)$is defined as follows:

1. if $\mathcal{P}$ is empty, then $S(\mathcal{P}, \mathcal{E}, s) = s$;
2. if $\mathcal{P} = \exists x.\mathcal{P}'$, and v is the truth value making $\mathcal{P}'.(\mathcal{E}|(s \cdot v))$ valid, then $S(\mathcal{P}, \mathcal{E}, s) = S(\mathcal{P}', \mathcal{E}, s \cdot v)$;
3. if $\mathcal{P} = \forall x.\mathcal{P}'$ then $S(\mathcal{P}, \mathcal{E}, s) = S(\mathcal{P}', \mathcal{E}, s \cdot 0) \circ S(\mathcal{P}', \mathcal{E}, s \cdot 1)$

Note that the value v in Point 2. exists by Definition 4. We now prove that computing $S(P, E, \emptyset)$ one obtains a sequence of assignments that is consistent with P, and such that each of its assignments satisfies E.

The second claim follows from an invariant: from the first call, and at each inner recursive call of $S(\mathcal{P}, \mathcal{E}, s)$, the formula $\mathcal{P}.(\mathcal{E}|s)$ is valid, hence it is valid also for leaf calls which yield the total assignments.

Regarding the first claim, we prove the following properties of S: for each call of $S(\mathcal{P}, \mathcal{E}, s)$, the sequence of assignments it generates contains $2^{|\mathrm{UNIVS}(\mathcal{P})|}$ assignments, starts with an assignment st for which $\pi_{\mathrm{UNIVS}(\mathcal{P})}(t) = 0 \cdots 0$, and ends with an assignment of the form sq for which $\pi_{\mathrm{UNIVS}(\mathcal{P})}(q) = 1 \cdots 1$. All of these properties are proved by induction on the number $|\mathrm{UNIVS}(\mathcal{P})|$. Then, Points 1. and 2. of Definition 2 immediately follow, and Point 3. is proved again by induction on $|\mathrm{UNIVS}(\mathcal{P})|$.

$\Leftarrow$ Let $s_1, \ldots, s_m$ be the consistent sequence of assignments. Let $b_1^i \cdots b_k^i$ denote the first k assignments of s_i. The proof is by induction, on the following statement: if $Q_{k+1} x_{k+1} \cdots Q_n x_n.(E|b_1^i \cdots b_k^i)$ are all valid, for every $i = 1, \ldots, m$, then $Q_k x_k \cdots Q_n x_n.(E|b_1^i \cdots b_{k-1}^i$ are all valid. The base case is for $k = n$, which is true because every assignment satisfies E. We work backwards on k, achieving the proof for $k = 0$. If $Q_k = \exists$, then the inductive claim is trivially true, since b_k^i is either 0 or 1. If $Q_k = \forall$, suppose $b_k^i = 0$. Let $u = |\mathrm{UNIVS}(Q_{k+1} x_{k+1} \cdots Q_n x_n)|$. Then since the sequence is consistent with P, the assignment s_j, with $j = i + 2^u$, is such that $b_1^i \cdots b_{k-1}^i = b_1^j \cdots b_{k-1}^j$, and $b_k^j = 1$. Since also $Q_{k+1} x_{k+1} \cdots Q_n x_n.(E|b_1^j \cdots b_k^j)$ is valid, the claim follows. □

We now prove the correspondence between paths falsifying the specification in the Kripke structure of the translation of a QBF, and consistent sequences of assignments all satisfying E.

Theorem 2 *Let $F = P.E$ be a QBF, and let S_F be the Kripke structure defined by the translation of F. Then*

the specification is false iff there exists a sequence of assignments, consistent with P and such that every assignment satisfies E.

Proof. $\Rightarrow$ If the specification is false, then there exists an infinite path satisfying $G(clauses.sat)$, that is, in every state the assignment to variables satisfies E. We now prove that the first $m = 2^{|\text{UNIVS}(P)|}$ states of the path define a consistent sequence of assignments.

We go through all points of Definition 2. Point 1. is true by hypothesis.

The initial state of the SMV model assigns 0 to every universal variable, hence the assignment of the initial state satisfies Point 2. of Definition 2.

For Point 3, let s_i and s_{i+1} be two states in the path. To avoid multiple notations, let also s_i and s_{i+1} denote the truth assignments obtained by taking the value of variable v in each module. If E is true, then clauses.sat is 1, hence the carry-in of the module corresponding to the most internal variable is 1. Modules for existential variables just pass the carry over to the next module, so modules for universal variables act as a standard binary counter (see [HR00]). This satisfies Condition 1. of Definition 1. Moreover, if $\pi_{1,\ldots,k}(\pi_{\text{UNIVS}(P)}(s_1)) = \pi_{1,\ldots,k}(\pi_{\text{UNIVS}(P)}(s_2))$, then carry-in of the modules 1–k must be 0. But then, the existential modules just replicate in s_{i+1} the value of v that was set in s_i, hence $\pi_{1,\ldots,k}(\pi_{\text{EXISTS}(P)}(s_1)) = \pi_{1,\ldots,k}(\pi_{\text{EXISTS}(P)}(s_2))$. This satisfies Condition 2. of Definition 1. In conclusion, $s_i \rightsquigarrow s_{i+1}$.

$\Leftarrow$ Follows easily from the definition of a consistent sequence of assignments. The infinite path goes from s_1 through s_m and then back to s_1.

□

Simply combining Theorem2 and Theorem1, we can now state our main theorem.

Theorem 3 *Given a QBF F, let P_F be the corresponding* SMV *Program. Then, the specification in P_F is false if and only if F is true.*

As for the size of the encoding, it is clear that modules `forall` and `exists` have constant size. The module `formula` has the same size of E, while `main` has size linear in the number of variables of F. The size of the specification is also constant (it does not depend on the QBF F). Moreover, it is easy to show that this encoding can be computed using additional logarithmic workspace.

Obviously this doesn't mean that the final expanded model would have size $O(poly(n, m))$. However, we are only interested in mapping a QBF into an SMV-specification of polynomial size. This result implies that model-checking in SMV is PSPACE-hard even for specifications of constant size.

6 Experimental Results

In this section we present the most important experimental results on randomly generated QBF instances. All instances are obtained using the 2QBF-5CNF FCL2 model [CSGG00], we now recall.

In the early QBF work by Cadoli et al. [CGS97], random kQBF instances were generated according to the *Fixed Clause Length* (FCL) model [SML96]. The FCL model for QBFs has the following parameters:

- the number k of distinct sets of propositional variables in formula $Q_1 X_1 \cdots \exists X_k \,.\, E(X_1, \ldots, X_k)$,
- the cardinalities $|X_1|, \ldots, |X_k|$,
- the number m of clauses in E,
- the number h of literals per clause.

In this model each formula is generated so that every clause contains h literals. If V is the set $X_1 \cup X_2 \cup \cdots \cup X_k$, then a clause is produced by randomly choosing h distinct variables in V and negating each one with probability 0.5. The FCL model for QBF directly extends the FCL model for SAT.

The model also constraints the m clauses to be different to each other, and that none of them includes both a literal and its complement. Moreover, to avoid generating trivially false kQBF instances, a clause cannot contain universally quantified variables only.

Gent and Walsh [GW99] noted that with the FCL model the probability of generating instances containing two clauses such that in each clause all variables except one are universally quantified and the remainig variabla also appears in the other clause with the opposite sign increases very quickly when m increases. Such instances are therefore false. This is why the FCL2 model has been introduced. In this model, if a clause contains a literal l and another clause contains $\neg l$, then one of the clauses is removed and replaced. FCL2 is similar to two models (named A and B) investigated by Gent and Walsh [GW99], the difference being that the selection criterion of FCL2 is more precise: for example, in model A a clause with less than two existential variables (which does not necessarily

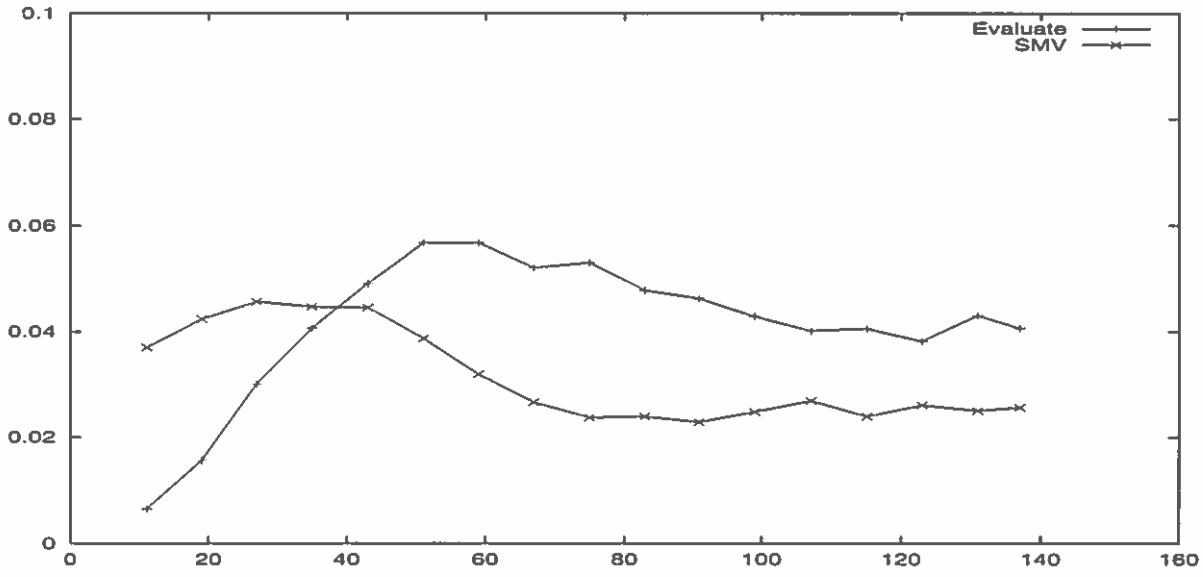


Figure 8: Evaluate vs. SMV (2QBF-5CNF, 14 variables, FCL2 model)

make the formula false) is disallowed. Extensive testing has been done on FCL2 [CSGG00].

Figure 8 reports the result of a comparison between direct QBF solving with Evaluate and solving by reduction to SMV. This test has been done with 14 variables (of which 7 are universally quantified and 7 are existentially quantified). Two facts are quite evident: first, the performance of the two algorithms are comparable, showing that SMV is a viable alternative to direct QBF solving; second, hard QBF instances remain hard after the translation to SMV, which implies that hard random SMV instances can be obtained from hard QBF instances by reduction. Finally, while the shape of the curves are more or less the same, their precise values are quite different. Namely, the peak generated by SMV is moved to the left, and the increase before the peak is less steep. All these observations confirm the usefulness of the proposed translation in practice. Significant differences in the relative efficiency of BDD-based and Davis-Putnam -based solvers for SAT problems have also been reported by Giunchiglia et. al. in [GNT01] and Vardi et al. in [CDSMA+00].

Figure 9 shows the comparison with formulae of ten variables. Some more experiments have been run using NuSMV instead of SMV. The results are reported in Figure 10, 11, 12. Experiments are still running at the time of this writing; other ones suggested by an anonymous reviewer are planned.

7 Conclusions

In this paper we have shown that the evaluation of a QBF can be reduced to model-checking of a simple (constant) formula over an SMV model. The theoretical implication of this result is that CTL and LTL model checking are PSPACE-hard in the size of the structure, when the structure is written using a practical language like the SMV input language. More importantly, the reduction can be exploited to generate challenging benchmarks for model-checking systems. This is useful, as recent results in the QBF literature [CSGG00, EETW00] show that it is possible to generate very hard instances for QBF in a controlled way.

A significant result of the performed test is that the overall QBF algorithm composed of the translation and the solution by SMV has computational properties comparable with those of direct QBF solvers. It is also interesting how some properties, like the position of the peak, that were believed to be intrinsic of the QBF problem, do not appear, suggesting that they are related to the specific solving algorithm.

Acknowledgments

The authors thank Enrico Giunchiglia and Alessandro Cimatti for interesting discussions on the topic of this paper. They also thank the anonimous reviewers for their useful suggestions. Work partially supported by the Italian Ministry for University and Research

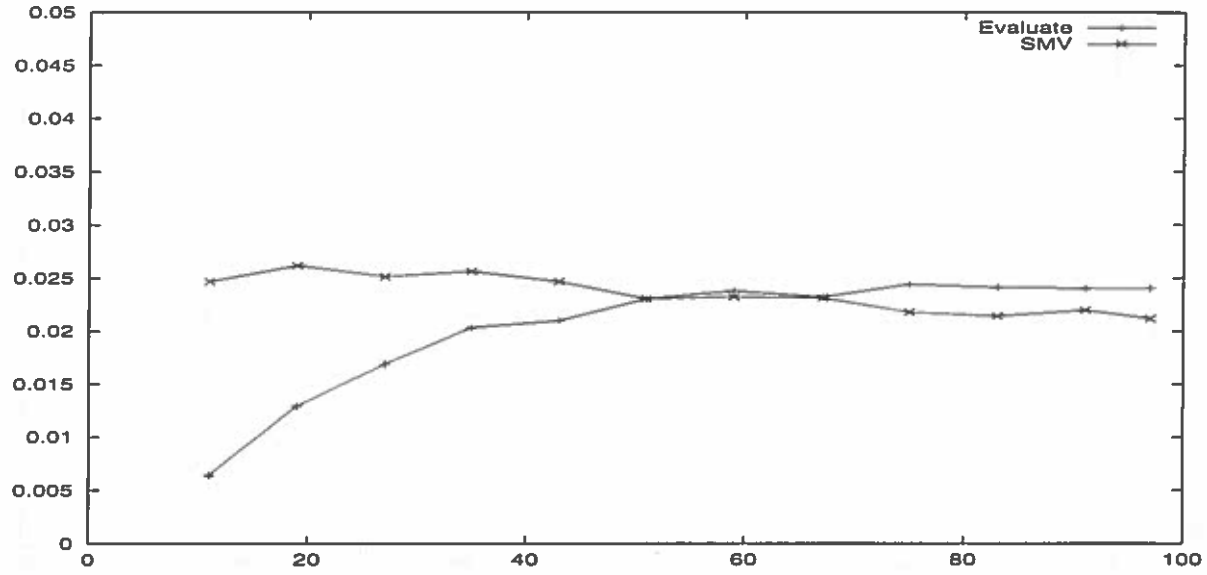


Figure 9: Evaluate vs. SMV (2QBF-5CNF, 10 variables, FCL2 model)

(project MOSES) and Italian National Research Council (project LAICO).

References

[CCGR00] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: A new symbolic model checker. *International Journal on Software Tools for Technology Transfer (STTT)*, 2(4):410–425, 2000.

[CCO+] Roberto Cavada, Alessandro Cimatti, Emanuele Olivetti, Marco Pistore, and Marco Roveri. Nusmv 2.0 user manual. Available on the Web at http://nusmv.irst.itc.it/NuSMV/userman/.

[CDSMA+00] C Coarfa, D Demopoulos, A. San Miguel Aguirre, D. Subramanian, and M Vardi. Random 3-sat: The plot thickens. In *Proc. of CP 2000*, pages 143–159, 2000.

[CGS97] Marco Cadoli, Andrea Giovanardi, and Marco Schaerf. Experimental analysis of the computational cost of evaluating quantified boolean formulae. In *Proc. of AI*IA '97*, number 1321 in LNAI, pages 207–218. Springer-Verlag, 1997.

[CSGG00] Marco Cadoli, Marco Schaerf, Andrea Giovanardi, and Massimo Giovanardi. An algorithm to evaluate quantified boolean formulae and its experimental evaluation. In Ian Gent, Hans van Maaren, and Toby Walsh, editors, *SAT2000 - Highlights of Satisfiability Research in the Year 2000*, pages 485–521. IOS Press, 2000. Will appear also on *Journal of Automated Reasoning*, special issue on *SAT2000*.

[DGV99] M. Daniele, F. Giunchiglia, and M. Vardi. Improved automata generation for linear temporal logic. In *Proc. of CAV'99*, volume 1633 of *LNCS*, pages 249–260. Springer-Verlag, 1999.

[DLL62] M. Davis, G. Logemann, and D. W. Loveland. A machine program for theorem proving. *Comm. of the ACM*, 5(7):394–397, 1962.

[EETW00] U. Egly, T. Eiter, H. Tompits, and S. Woltran. Solving advanced reasoning tasks using quantified boolean formulas. In *Proc. of AAAI 2000*, pages 417–422, 2000.

[FMS00] R. Feldmann, B. Monien, and S. Schamberger. A distributed algorithm to evaluate quantified boolean formulae. In *Proc. of AAAI 2000*, pages 285–290, 2000.

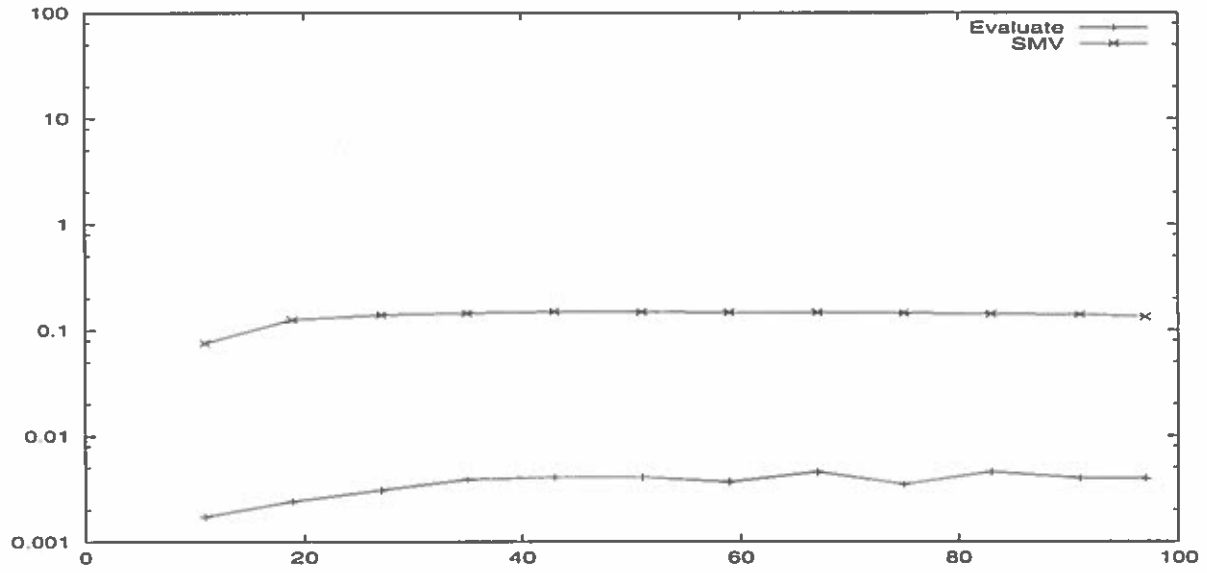


Figure 10: Evaluate vs. NuSMV (2QBF-5CNF, 10 variables, FCL2 model)

[GNT01] E. Giunchiglia, M. Narizzano, and A. Tacchella. An analysis of backjumping and trivial truth in quantified boolean formulas satisfiability. In *Proceedings of the Italian Conference on Artificial Intelligence*, pages 25–28, 2001.

[GW99] I. P. Gent and T. Walsh. Beyond NP: the QSAT phase transition. In *Proc. of AAAI'99*, 1999.

[HR00] M. Huth and M. Ryan. *Logic in Computer Science. Modelling and reasoning about systems.* Cambridge University Press, 2000.

[McM93] Kenneth L. McMillan. *Symbolic Model Checking.* Kluwer Academic Publisher, 1993.

[Rin99] J. Rintanen. Improvements to the evaluation of quantified boolean formulae. In *Proc. of IJCAI'99*, pages 1192–1197, July-August 1999.

[SC85] A. Prasad Sistla and E. M. Clarke. The complexity of propositional linear temporal logic. *J. of the ACM*, 32(3):733–749, 1985.

[SML96] B. Selman, D. Mitchell, and H. Levesque. Generating Hard Satisfiability Problems. *Artificial Intelligence*, 81:17–29, 1996.

[Sto76] L. J. Stockmeyer. The polynomial-time hierarchy. *Theor. Comp. Sci.*, 3:1–22, 1976.

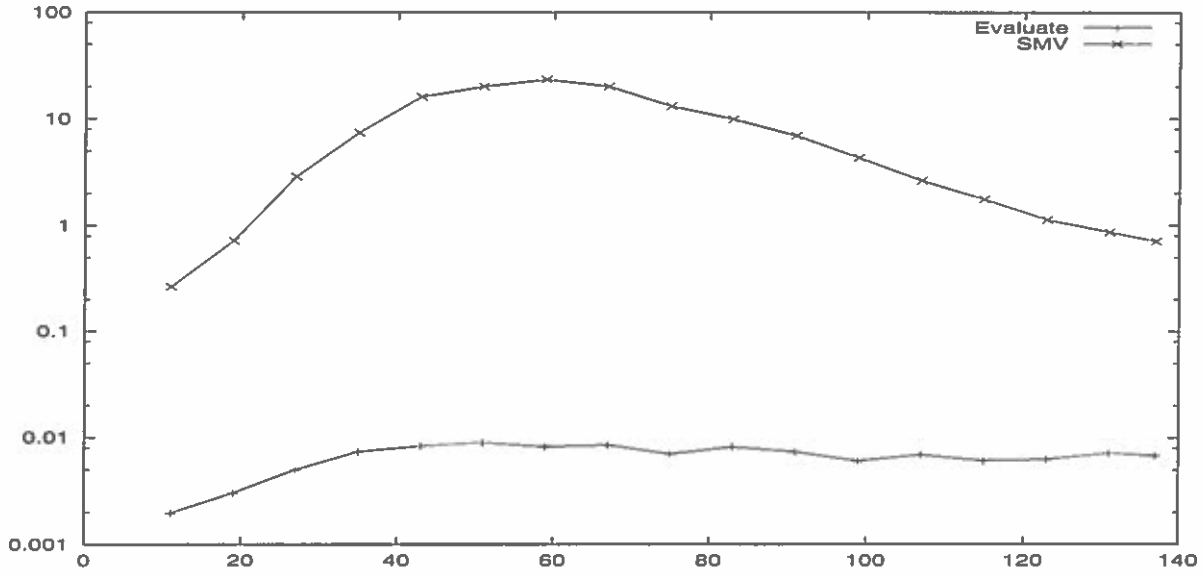


Figure 11: Evaluate vs. NuSMV (2QBF-5CNF, 14 variables, FCL2 model)

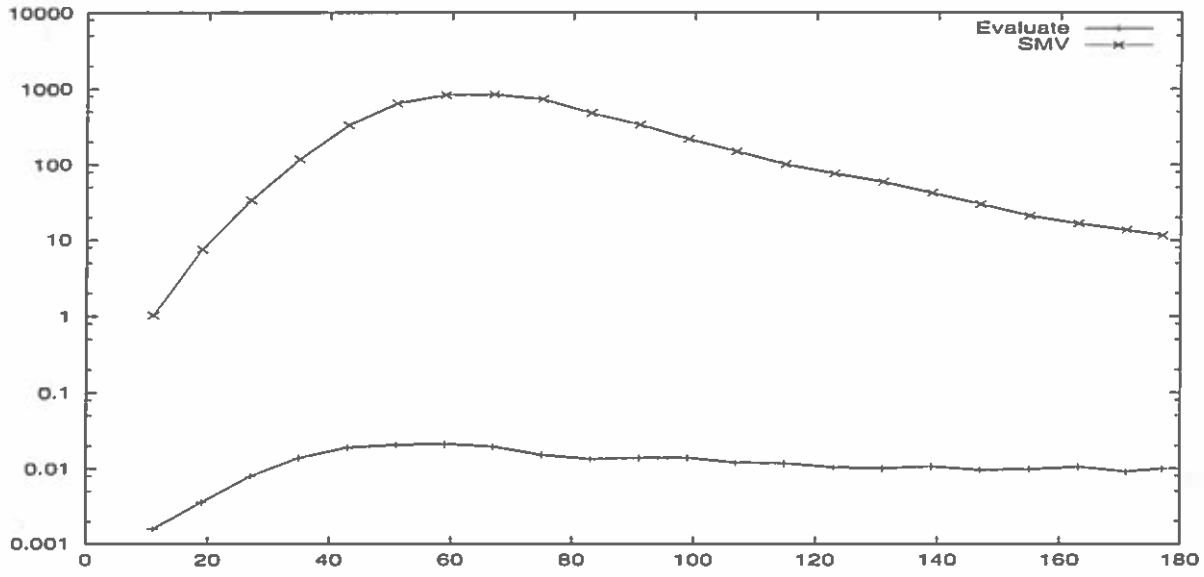


Figure 12: Evaluate vs. NuSMV (2QBF-5CNF, 18 variables, FCL2 model)

Knowledge Representation and Planning (Joint Session with AIPS)

Reasoning about Actions and Planning in LTL Action Theories

Diego Calvanese, Giuseppe De Giacomo
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
lastname@dis.uniroma1.it

Moshe Y. Vardi
Department of Computer Science
Rice University, P.O. Box 1892
Houston, TX 77251-1892, U.S.A.
vardi@cs.rice.edu

Abstract

In this paper, we study reasoning about actions and planning with incomplete information in a setting where the dynamic system is specified by adopting Linear Temporal Logic (LTL). Specifically, we study: (i) reasoning about action effects (i.e., projection, historical queries, etc.), in such a setting; (ii) when actions can be legally executed, assuming a non-prescriptive approach, where executing an action is possible in a given situation unless forbidden by the system specification; (iii) the problem of finding conformant plans for temporally extended goals that consist of arbitrary LTL formulas, thus allowing for expressing sophisticated dynamic requirements. For each of these problems we establish techniques and characterize the computational complexity. For the last two problems we make use of a second-order variant of LTL.

1 INTRODUCTION

Linear Temporal Logic (LTL) is a linear-time temporal logic which has been widely used for specifying and verifying properties of dynamic systems, such as safety, liveness, fairness, etc. [26, 13, 39]. In this paper, we study reasoning about actions and planning in a setting where we have incomplete information on the dynamic system and our knowledge on it is represented in LTL. This means that we represent the behavior of the system as a set of sequences of situations, where transitions from one situation to the next are caused by actions. In particular, we describe the system by introducing a set of atomic facts, called fluents, whose truth value changes as the system evolves, and by specifying through a logic (LTL, in our case) the effects of actions on such a set of facts. Such an approach is shared by several proposals for reasoning about actions, e.g., [4, 24, 12]. Here, however, we do not focus on specific action theories for the dynamic system. Instead, we allow the system specification to be an arbitrary LTL formula.

First, we address reasoning about action effects in the above setting. In particular, we focus on problems of the following form: given a finite sequence of actions, determine whether a certain property holds (which include projections and historical queries [28]).

Next, we address executability of actions in a linear-time setting. We observe that linear-time logics cannot be used for prescriptively asserting which actions are possible in a given situation. This is different from branching-time formalisms, such as the situation calculus [28], the fluent calculus [34], dynamic logics [10], and branching-time temporal logics [7, 9], where executability can be expressed directly in the language. There are some workarounds to this problem in the linear-time setting, but they involve action theories of a specific form, see [30] for a discussion. Here, instead, we follow a non-prescriptive approach (similar in the spirit to [16, 22, 3]): executing an action is possible in a given situation, unless forbidden by the system specification. We formalize and study such a notion in the context of LTL.

Finally, we address the problem of finding *conformant plans* for *temporally extended goals* in dynamic systems in the presence of incomplete information on the initial situation and on the full effects of actions. Conformant plans are sequences of actions that are guaranteed to fulfill the goal specification, even when we have incomplete information on the dynamic system in which the plan is to be executed [14, 17, 33, 28, 5]. Temporally extended goals are goals that specify acceptable sequences of states [1]. They subsume the usual goals expressing "reachability" of desired conditions, as well as generalized goals, such as "don't-disturb" and "re-

store" requirements [41]. More generally, they allow for expressing complex temporal properties typically used in the specification of processes [13, 39]. Planning for temporally extended goals has been studied in [1, 6, 2, 21], where complete information is assumed, and in [11], where the system is specified as a deterministic transition system, except for the initial situation, on which incomplete knowledge is assumed. Related kinds of goals, expressed in CTL, have been investigated in [25]. We observe that CTL [13] is a branching temporal logic whose expressive power is incomparable to that of LTL. Here we address conformant planning for temporally extended goals specified by means of arbitrary LTL formulas.

Observe that, as we deal with goals expressing general temporal properties, even sequential plans may in fact involve loops, since goals of this form may require *infinite executions*. Consider, for example, a plan to satisfy the following requirement: whenever certain triggering conditions are met within a finite (but undetermined) number of steps, a specified state of affairs must be brought about in which the triggering conditions are met again.

For each of the above three problems, i.e., reasoning about action effects, reasoning about executability of actions, and conformant planning, we establish techniques and characterize the computational complexity. To formally capture the last two problems and to derive reasoning procedures, we make use of a second-order variant of LTL.

In this paper we focus on LTL as the linear-time formalism. However, all the results presented here can be immediately rephrased in terms of more expressive linear-time formalisms, such as μLTL, which is able to express any ω-regular property [37].

2 PRELIMINARIES

We introduce the temporal logic LTL and discuss its relationship to Büchi automata on infinite strings. We also introduce an extension of LTL with second-order quantification.

2.1 LINEAR TEMPORAL LOGIC (LTL)

Linear Temporal Logic (LTL) was originally proposed in Computer Science as a specification language for concurrent programs [26]. *Formulas* of LTL are built from a set $\mathcal{P}$ of propositional symbols and are closed under the boolean operators, the unary temporal operators $\bigcirc$, $\Diamond$, and $\Box$, and the binary temporal operator $\mathcal{U}$. Intuitively, $\bigcirc\varphi$ says that φ holds at the *next* instant, $\Diamond\varphi$ says that φ will *eventually* hold at some future instant, $\Box\varphi$ says that from the current instant on φ will *always* hold, and $\varphi\,\mathcal{U}\,\psi$ says that at some future instant ψ will hold and *until* that point φ holds. In fact, it is sufficient to consider as temporal operators only $\bigcirc$ and $\mathcal{U}$, since $\Diamond\varphi$ can be viewed as an abbreviation for $\mathbf{true}\,\mathcal{U}\,\varphi$, and $\Box\varphi$ as an abbreviation for $\neg\Diamond\neg\varphi$. We additionally use the standard boolean abbreviations $\vee$ (or) and $\rightarrow$ (implies).

The semantics of LTL is given in terms of interpretations over a *linear structure*. Without loss of generality and for simplicity of presentation, we use the natural numbers $\mathbf{N}$ as the linear structure. Hence, for an instant $i \in \mathbf{N}$, the successive instant is $i+1$. An *interpretation* is a function $\pi : \mathbf{N} \rightarrow 2^{\mathcal{P}}$, which assigns to each element of $\mathcal{P}$ a truth value at each instant $i \in \mathbf{N}$. For an interpretation π, we inductively define when an LTL formula φ *is true* at an instant $i \in \mathbf{N}$, in symbols $\pi, i \models \varphi$, as follows:

- $\pi, i \models p$, for $p \in \mathcal{P}$ iff $p \in \pi(i)$.
- $\pi, i \models \neg\varphi$ iff not $\pi, i \models \varphi$.
- $\pi, i \models \varphi \wedge \varphi'$ iff $\pi, i \models \varphi$ and $\pi, i \models \varphi'$.
- $\pi, i \models \bigcirc\varphi$ iff $\pi, i+1 \models \varphi$.
- $\pi, i \models \varphi\,\mathcal{U}\,\varphi'$ iff for some $j \geq i$, we have that $\pi, j \models \varphi'$ and for all k, $i \leq k < j$, we have that $\pi, k \models \varphi$.

A formula φ *is true* in π, in notation $\pi \models \varphi$, if $\pi, 0 \models \varphi$. A formula φ is *satisfiable* if it is true in some interpretation, and is *valid*, if it is true in every interpretation.

Theorem 1 [31] *Satisfiability (and validity) for* LTL *formulas are PSPACE-complete.*

2.2 BÜCHI AUTOMATA AND CORRESPONDENCE WITH LTL

There is a tight relation between LTL and Büchi automata on infinite words (see e.g., [39]).

Given a finite nonempty alphabet Σ, an *infinite word* is an element of Σ^ω, i.e., an infinite sequence $a_0 a_1 \cdots$ of symbols from Σ. A *Büchi automaton* [36] is a tuple $A = (\Sigma, S, S_0, \rho, F)$ where:

- Σ is the alphabet of the automaton.
- S is the finite set of states.
- $S_0 \subseteq S$ is the set of initial states.
- $\rho : S \times \Sigma \rightarrow 2^S$ is the transition function of the automaton (the automaton does not need to be deterministic).

- $F \subseteq S$ is the set of accepting states.

The *input words* of A are infinite words $a_0 a_1 \cdots \in \Sigma^\omega$. A *run* of A on an infinite word $a_0 a_1 \cdots$ is an infinite sequences of states $s_0 s_1 \cdots \in S^\omega$ such that $s_0 \in S_0$ and $s_{i+1} \in \rho(s_i, a_i)$. A run r is *accepting* iff $lim(r) \cap F \neq \emptyset$, where $lim(r)$ is the set of states that occur in r infinitely often. In other words, a run is accepting if it gets into F infinitely many times, which in turn means, being F finite, that there is at least one state $s_f \in F$ that is visited infinitely often. The *language* accepted by A, denoted by $L(A)$, is the set of words for which there is an accepting run. Nondeterministic Büchi automata are closed under intersection and complement [36]. More precisely, for two Büchi automata A_1 and A_2, the a number of states of the automaton accepting $L(A_1) \cap L(A_2)$ is polynomial in the number of states of A_1 and A_2 [40], while the number of states of the automaton accepting $\Sigma^\omega \setminus L(A_1)$ is exponential in the number of states of A_1.

The *nonemptiness* problem for an automaton is to decide, given an automaton A, whether $L(A) \neq \emptyset$, i.e., whether the automaton accepts at least one word.

Theorem 2 [40] *Nonemptiness of Büchi automata is NLOGSPACE-complete.*

The nonemptiness algorithm in [40] actually returns a witness for nonemptiness, which is a finite prefix followed by a cycle.

Both automata on infinite words and linear time logics, such as LTL, are widely used in verification to specify properties of dynamic systems [20, 40, 38]. The two formalisms can be put in a tight correspondence by considering as alphabet of the automaton the set $2^{\mathcal{P}}$ of propositional interpretations of the propositional variables in $\mathcal{P}$. Hence, an infinite word over the alphabet $2^{\mathcal{P}}$ accepted by an automaton can be viewed as an interpretation of an LTL formula over $\mathcal{P}$.

Theorem 3 [40] *For every* LTL *formula φ one can effectively construct a Büchi automaton A_φ whose number of states is at most exponential in the length of φ and such that $L(A_\varphi)$ is the set of models of φ.*

2.3 QUANTIFIED LINEAR TEMPORAL LOGIC (QLTL)

We make also use of an extension of LTL by second-order quantifiers over propositions, called Quantified Linear Temporal Logic (QLTL) [32]. Formally, formulas of QLTL are built using the operators of LTL plus an operator for existential quantification over propositions, i.e., $\exists p.\varphi(p)$, where p is a proposition variable and $\varphi(p)$ is a QLTL formula in which p occurs free. We use also $\forall p.\varphi(p)$ as an abbreviation for $\neg\exists p.\neg\varphi(p)$. The semantics of such an operator is defined as follows

- $\pi, i \models \exists p.\varphi$ iff there is some π' that agrees with π except for the interpretation of proposition p, and such that $\pi', i \models \varphi$.

Observe that the two interpretations π and π' may disagree on the interpretation of p at any time point, not only the initial one.

Every QLTL formula can be put in *prefix normal form*, i.e., written in the form

$$Q_1 p_1 . Q_2 p_2 . \cdots Q_h p_h . \varphi$$

where each Q_i is either $\forall$ or $\exists$, and φ is a (quantifier-free) LTL formula. If Q_1 is $\exists$ and there are $k-1$ alternations of quantifiers, we say that the formula is a Σ_k^{QLTL} formula. If Q_1 is $\forall$ and there are $k-1$ alternations of quantifiers, we say that the formula is a Π_k^{QLTL} formula.

The following complexity characterization of satisfiability for QLTL formulas was given in [32] (for a positive integer k, k-EXPSPACE denotes the set of languages accepted by a Turing machine with space bounded by $2^{2^{\cdots^{2^n}}}$, where the height of the tower is k, and n is the size of the input tape).

Theorem 4 [32] *Satisfiability for Σ_{k+1}^{QLTL} and for Π_k^{QLTL} formulas, with $k \geq 1$, is k-EXPSPACE-complete.*

The result is obtained by reducing satisfiability of a QLTL formula φ to non-emptiness of Büchi automata. Intuitively, one first builds the Büchi automaton for the LTL matrix of φ. Then, existential quantification is handled by projecting out the existentially quantified propositions from the automaton, while universal quantification requires a complementation [19], which gives rise to an exponential blow-up in the number of states of the automaton.[1]

3 REASONING ABOUT ACTIONS USING LTL

We may characterize the behavior of a dynamic system by a set of evolutions, each of which can be represented as a sequence of *situations* [28]. Transitions from one situation to the next are caused by actions. If

[1] [15] describes a symbolic implementation of the complementation construction described in [19].

we have complete information on the current situation and complete information on the effects of actions, we are able to determine the actual evolution of the system corresponding to a given sequence of actions. Typically, however, we have incomplete information both on the current situation and on the actual effects of actions. So, given a sequence of actions, we will only be able to isolate a set of possible evolutions, one of which is the actual one. It follows that, if we want to check whether a certain dynamic property holds, we need to check it for every possible evolution.

Following a methodology typical of the literature on reasoning about actions in AI (cf. [28, 30, 29, 16, 35]) we specify a dynamic system by introducing a set of atomic facts, here called *fluents*, whose truth value changes as the system evolves, and by specifying, through a logical formalism, the *effects* of *actions* on such a set of facts. Then, given a dynamic property, we can use logical inference to check whether the property holds.

3.1 STRUCTURAL REQUIREMENTS

Specifically, we adopt as logical formalism LTL. Since LTL does not provide us a direct notion of action, we use propositions to denote them. Hence we consider two separate sets of atomic propositions:

- *fluents* $\mathcal{F}$, which are propositions that denote atomic facts on the current situation;
- *actions* $\mathcal{A}$, which are special propositions denoting that an action has just been performed.

We describe the dynamic system by means of a conjunction of a finite set of LTL formulas. To suitably model actions we always add as conjuncts to the specification of the dynamic system the following LTL formulas:

$$\Box(\bigvee_{a \in \mathcal{A}} a)$$

to specify that one action must be performed in order to get to a new situation, and

$$\Box(\bigwedge_{a \in \mathcal{A}} (a \rightarrow \bigwedge_{b \in \mathcal{A}, b \neq a} \neg b))$$

to specify that a single action at a time can be performed.

In this way, a unique proposition $a \in \mathcal{A}$ holds in each situation, and it specifies the action a that has been just performed to get to that situation[2]. We include in $\mathcal{A}$ a proposition a_d, representing a dummy action with no effects. In the initial situation, where no actual action has been performed yet, we require for uniformity that a_d holds.

[2]In fact, the approach can be easily extended to deal with concurrent atomic actions.

3.2 SPECIFYING EFFECTS

Apart from such structural requirements on modeling actions, we allow for (finite) sets of arbitrary LTL formulas in specifying the dynamic system. A simple way to specify the dynamic system is as follows:

- We describe the *initial situation* as an arbitrary propositional formula φ_{init} involving only fluents.
- We specify *effects of actions* by means of formulas of the form

  $$\Box(\varphi \rightarrow \bigcirc(a \rightarrow \psi))$$

  where ψ and φ are arbitrary propositional formulas involving only fluents. Such a formula says that executing the action a under the conditions denoted by φ brings about the conditions denoted by ψ.[3]
- We may also specify *state constraints* by means of formulas of the form

  $$\Box\phi$$

  where ϕ is a propositional formula involving only fluents.

Observe that with this formalization we may have incomplete information on the initial situation. Moreover we may have incomplete information on the effects of an action, i.e., even if we know the truth-values of all the fluents in a given situation we may not know their value after the execution of an action.

If we consider the above formalization to be too liberal and we are willing to completely specify the effects of actions, then we can use, for example, LTL formulas that correspond to Reiter's *successor state axioms* [28] (which also provide a solution to the frame problem). Namely

$$\Box(\bigcirc F \equiv \bigvee_{a}(\varphi_a^+ \wedge \bigcirc a) \vee (F \wedge \bigwedge_{b}(\neg\varphi_b^- \vee \bigcirc\neg b)))$$

where F is a fluent, the a's are those actions that under the circumstances described by the propositional formulas φ_a^+, involving only fluents, make F become true,

[3]Note that the formula $\Box(\varphi \rightarrow \bigcirc(a \rightarrow \varphi))$ corresponds to a frame axiom expressing that φ does not change performing a.

and the b's are those actions that under the circumstances described by the propositional formulas φ_b^-, involving only fluents, make F become false. Hence the formula above expresses that F is true next if and only if either one of the a's is executed and φ_a^+ is currently true, or F is currently true, and none of the b's, such that φ_b^- is currently true, is executed.

If instead we think that the above formalization is too restricted, then LTL allows us also to express very loose effect specifications, such as "continuing to chop a tree sooner or later makes it fall down": $(\Diamond\Box\neg chop) \vee \Diamond falls\_down$, i.e., there cannot be a sequence of situations in which the tree is chopped infinitely often but it does not fall down.

Finally, LTL allows for expressing in a natural way several forms of *narratives* [29, 30, 27]. Indeed, an LTL formula may express that certain actions occur, or certain conditions are brought about, according to specified temporal patterns. For example, "after three steps Mary arrives to the airport, then, eventually, she boards the plane", can be expressed as $\bigcirc\bigcirc\bigcirc(arrives\_airport \wedge \Diamond boards\_plane)$.

3.3 REASONING ABOUT ACTIONS EFFECTS IN LTL

The discussion above shows that LTL is well suited to describe dynamic systems wrt action effects. In this paper, however, we do not focus on specific ways to formalize the dynamic system. Instead, as mentioned, we allow for any LTL formula as a description of the dynamic system, as long as the structural requirements are enforced.

We can use LTL validity to reason about action effects, i.e., to solve problems of the following form: given a finite sequence of actions, determine whether a certain property holds [28]. Let Γ be the formula describing the dynamic system, and let us introduce the formula

$$\begin{aligned} &Occurs(a_0 \cdots a_k, rs) \doteq \\ &\quad (a_0 \wedge \bigcirc(a_1 \wedge \bigcirc(\cdots \bigcirc(a_k \wedge rs) \cdots))) \wedge \\ &\quad \Box(rs \rightarrow \bigcirc\Box\neg rs) \end{aligned}$$

which expresses that the sequence of actions $a_0 \cdots a_k$ occurs, resulting in a situation denoted by the new proposition rs (first conjunct), and that rs is true only once (second conjunct). Note that rs acts as a marker for the situation resulting by executing $a_0 \cdots a_k$. The *projection problem* (cf. [28]), "does the property φ hold after the execution of the sequence of actions $a_0 \cdots a_k$?", can be solved by checking the validity of

$$\Gamma \rightarrow (Occurs(a_0 \cdots a_k, rs) \rightarrow \Box(rs \rightarrow \varphi))$$

Also, *Historical queries* (cf. [28]) of the form "does the property φ always hold over the duration of the sequence of actions $a_0 \cdots a_k$?" can be answered by checking the validity of

$$\Gamma \rightarrow (Occurs(a_0 \cdots a_k, rs) \rightarrow \Box((\Diamond rs) \rightarrow \phi))$$

Similarly, historical queries of the form "does the property φ hold at some point over the duration of the sequence of actions $a_0 \cdots a_k$?" can be answered by checking the validity of

$$\Gamma \rightarrow (Occurs(a_0 \cdots a_k, rs) \rightarrow \Diamond(\varphi \wedge \Diamond rs))$$

Since validity in LTL is PSPACE-complete, we have that reasoning about action effects of a finite sequence of actions is PSPACE-complete as well. More formally we can state the following theorem.

Theorem 5 *Given an* LTL *formula* Γ *specifying a dynamic system, and an* LTL *formula* Φ *specifying a dynamic property, deciding whether* $\Gamma \rightarrow \Phi$ *is PSPACE-complete.*

4 LEGAL ACTION SEQUENCES

A fundamental question that arises is what actions are actually allowed at each given point. Often, in formalisms for reasoning about actions, one adopts a *prescriptive* approach, by specifying (explicitly or implicitly[4]) the circumstances under which an action is allowed at a given situation [28].

In LTL, on the other hand, we cannot express directly in the language that a certain action *is possible*. Hence we are forced to adopt a non-prescriptive approach, i.e, it is always possible to execute an action, unless it contradicts the system specification. However, one should be careful in adopting this notion since LTL does not adequately capture causality [22] in the presence of incomplete information.

Consider for example the formula $\Box((\bigcirc a) \rightarrow F)$, which says that, if action a is performed next, then the fluent F must be true now. While this is a logically meaningful sentence, it is problematic in defining allowable actions[5]. In principle, the actual state of the world corresponds to a certain truth assignment to the fluents. An action is *executable* if performing it does not

[4] Possibly involving a solution to the *qualification problem* [23, 34].

[5] Observe that, if we try to interpret the above formula causally, we get a quite counterintuitive interpretation: performing a has the effect of making F true in the situation preceding the execution of a, i.e., a would have an effect on the past.

contradict such a truth assignment. If in the actual state of the world F is true, then we can actually execute a, while if F is false, we cannot. Now, in general, we have only partial information on the current state of the world. So we need to ensure executability of actions whatever the current state of the world actually is. In our example, if we do not know whether F is true in the current situation, then we should not ask to execute a, since this may not be possible.

This difficulty is shared by virtually all linear-time formalisms, including those developed for reasoning about actions, such as the Event Calculus. The typical way to overcome this problem is to constrain the system specification so that the *principle of directionality* is fulfilled: "information about a given time is deductively independent from information about a later time", see [30] for a detailed discussion. This requires to choose special forms of logical theories for describing the system.

Here instead, we study the case where the system specification Γ itself is not constrained in any way, and introduce a notion of *legality* of a sequence of actions wrt Γ, to characterize the sequences of actions that are allowed. We say that a sequence $a_0 \cdots a_m$ of actions and a sequence $\sigma_0 \cdots \sigma_n$ of truth assignments to the fluents in $\mathcal{F}$ are *consistent with* Γ if there exists a model of Γ whose interpretation of the actions in the first m instants coincides with $a_0 \cdots a_m$ and whose interpretation of the fluents in the first n instants coincides with $\sigma_0 \cdots \sigma_n$. An infinite sequence of actions $a_0 a_1 \cdots$ is *consistent with* Γ if there exists a model of Γ whose interpretation of the actions coincides with $a_0 a_1 \cdots$. We say that an action a_{k+1} is *legal after the sequence of actions* $a_0 \cdots a_k$ if for all sequences $\sigma_0 \cdots \sigma_k$ of truth assignments to the fluents in $\mathcal{F}$, if $a_0 \cdots a_k$ and $\sigma_0 \cdots \sigma_k$ are consistent with Γ, then also $a_0 \cdots a_k a_{k+1}$ and $\sigma_0 \cdots \sigma_k$ are consistent with Γ. The sequence a_0 is trivially legal, as long as Γ is satisfiable[6]. A finite sequence of actions $a_0 \cdots a_k$ is *legal* if a_i is legal after $a_0 \cdots a_{i-1}$, for $i = 1 \ldots k$. An infinite sequence of actions $a_0 a_1 \cdots$ is *legal* if it is consistent with Γ, and if for all $k > 0$, a_k is legal after $a_0 \cdots a_{k-1}$.

The notion of legality cannot be expressed in an obvious way in LTL. However, in order to check the legality of a given finite sequence of actions wrt a system specification Γ, we can apply directly the definition of legality for finite sequences introduced above. This allows us to reduce the problem to a finite (although exponential) number of LTL satisfiability checks. It is easy to see that this provides us a PSPACE upper bound in the size of Γ and the length of the sequence.

Next we turn to infinite sequences, and we show that legality can be captured in QLTL. Let us introduce the formulas

$$\begin{aligned}
Point(now) &\doteq \Diamond now \wedge \Box(now \rightarrow \bigcirc\Box\neg now) \\
EqUntil(\vec{x}, \vec{y}, now) &\doteq \Box((\Diamond now) \rightarrow \vec{x} \equiv \vec{y}) \\
EqNext(\vec{x}, \vec{y}, now) &\doteq \Box(now \rightarrow \bigcirc(\vec{x} \equiv \vec{y}))
\end{aligned}$$

where now is a proposition that acts as a marker, $\vec{x}$ and $\vec{y}$ are tuples of variables, one for each action (fluent), and $\vec{x} \equiv \vec{y}$ stands for the conjunction of equivalences among corresponding components of the two tuples. The formula $Point(now)$ expresses that now holds at a single time point. $EqUntil(\vec{x}, \vec{y}, now)$ expresses that $\vec{x}$ and $\vec{y}$ coincide at every time point until the one where now holds. $EqNext(\vec{x}, \vec{y}, now)$ expresses that $\vec{x}$ and $\vec{y}$ coincide at the time point following the one where now holds.

Let us use $\Gamma(\vec{a}, \vec{f})$ to denote the system specification Γ in which we have explicited all the actions $\vec{a}$ and all the fluents $\vec{f}$ as parameters.

Then we can capture the notion of legality after a sequence of actions by means of the QLTL formula

$$\begin{aligned}
&LegalNext(\Gamma, \vec{a}, now) \doteq \\
&\quad Point(now) \wedge \\
&\quad \forall \vec{a}_1 . \forall \vec{f}_1 . \Gamma(\vec{a}_1, \vec{f}_1) \wedge EqUntil(\vec{a}_1, \vec{a}, now) \\
&\qquad\qquad \rightarrow \exists \vec{a}_2 . \exists \vec{f}_2 . \Gamma(\vec{a}_2, \vec{f}_2) \wedge \\
&\qquad\qquad\qquad EqUntil(\vec{a}_2, \vec{a}_1, now) \wedge \\
&\qquad\qquad\qquad EqUntil(\vec{f}_2, \vec{f}_1, now) \wedge \\
&\qquad\qquad\qquad EqNext(\vec{a}_2, \vec{a}, now)
\end{aligned}$$

Intuitively, such a formula expresses that for every interpretation of the actions and fluents satisfying Γ that agrees with $\vec{a}$ till now (but possibly differs in the fluents), there exists a further interpretation satisfying Γ that agrees both in the actions and the fluents with the first one till now, in which the action performed next is the one selected by $\vec{a}$.

Finally, we can characterize legal infinite sequences of actions by means of the QLTL formula:

$$Legal(\Gamma, \vec{a}) \doteq \exists \vec{f} . \Gamma(\vec{a}, \vec{f}) \wedge \forall now . LegalNext(\Gamma, \vec{a}, now)$$

Such a formula expresses that the sequence of actions resulting from the interpretation of $\vec{a}$ is consistent with Γ, and that every prefix of such a sequence continues next with a legal action.

Theorem 6 *An infinite sequence of actions $a_0 a_1 \cdots$ is legal wrt an LTL system specification Γ iff there exists an interpretation π interpreting the actions $\vec{a}$ according to $a_0 a_1 \cdots$ and such that $\pi \models Legal(\Gamma, \vec{a})$.*

[6] Recall that in our formalization a_0 must be the dummy action a_d.

Observe that, in defining $Legal(\Gamma, \vec{a})$, one alternation of second-order quantifiers is required. This is an indication that reasoning on legality is generally quite hard. Indeed, if we put $Legal(\Gamma, \vec{a})$ in prefix normal form, we get a Π_2^{QLTL} formula. Hence, considering Theorem 4 we get:

Theorem 7 *Checking the existence of an infinite sequence of actions that is legal wrt an* LTL *system specification Γ can be done in 2-EXPSPACE.*

Theorem 4 gives us in fact a constructive method to check the existence of a legal infinite sequence of actions using automata theoretic techniques. In particular, we start from $Legal(\Gamma, \vec{a})$ in prefix normal form. We construct a Büchi automaton A_1 corresponding to the matrix of such a formula. The automaton A_1 is exponential in the size of the matrix, and hence of Γ. Next we project out the existentially quantified variables from A_1, getting an automaton A_2. To deal with universal quantification, we complement A_2, obtaining an automaton A_3 of double exponential size, project out the universal quantified variables from A_3, and then complement again, obtaining an automaton A_4. The automaton A_4 would be of triple exponential size in Γ. However, we can do the last complementation on the fly while checking for non-emptiness, thus obtaining the 2-EXPSPACE upper bound.

It turns out that the bound in the previous theorem is actually tight.

Theorem 8 *Checking the existence of an infinite sequence of actions that is legal wrt an* LTL *system specification Γ is 2-EXPSPACE-hard.*

Proof sketch: Recall that satisfiability of Π_2^{QLTL} formulas is 2-EXPSPACE-complete. The lower bound proof in [32] is a reduction from Turing machines that require doubly exponential space. It cannot, however, be easily adapted to our setting. The difficulty is that in our problem Γ, which can be used to encode the Turing machine, appears on both the left-hand side and the right-hand side of an implication. The main difference between the two sides is the occurrence of the additional action described by *EqNext*. The key idea of our lower-bound proof is the introduction of a special action $a_{illegal}$ that, on one hand, cannot occur in a legal sequence of actions, while, on the other hand, guarantees that a finite sequence of actions in which $a_{illegal}$ occurs is consistent with Γ. As an example, suppose that Γ is the formula $\Diamond a_{illegal} \rightarrow p_0$. A finite sequence of actions in which $a_{illegal}$ occurs is consistent with Γ, since we can take p_0 to be true at time 0. On the other hand, the action $a_{illegal}$ cannot be legal, since it implies backward directionality.

To encode a Turing machine that requires doubly exponential space, we intend the sequence of actions to represent an accepting run of such a machine. Such a run consists of a sequence of configurations, each one of length 2^{2^n} (Γ has to be of length polynomial in n). To say that such a sequence of configurations is a proper computation of the machine, we need to "point" to cells that are distance 2^{2^n} apart and say that they are properly related. Such "pointing" is accomplished in [32] via second-order universal quantification.

Here we have implicit universal quantification over the sequence $\sigma_0, \ldots, \sigma_k$ of truth assignment to fluents in the definition of legality. In particular we have special fluents p_{first} and p_{last} that can hold at at most one point (this is ensured by Γ). The intention is for p_{last} to hold at point k. This is accomplished by requiring in Γ that $\Box(p_{last} \rightarrow \bigcirc\bigcirc a_{illegal})$. We know that in a legal sequence a_i cannot be $a_{illegal}$, for $1 \leq i \leq k+1$, so if p_{last} holds it must be at point k. We can now distinguish between the left-hand side and the right-hand side in the definition of legality. In the left-hand side, a_{k+1} is not yet defined, so it can be $a_{illegal}$. On the right-hand side a_{k+1} is an action in a legal sequence, so it cannot be $a_{illegal}$. Now we can use the left-hand side to say that p_{first} and p_{last} holds at points that are 2^{2^n} apart, and we use the right-hand side to require that the actions in these points are properly related. Further details are left to the full paper. ■

Hence, verifying existence of a sequence of actions that is legal wrt a system specification is 2-EXPSPACE-complete. To the best of our knowledge this is the hardest natural problem known for LTL.

Although not optimal from the point of view of computational complexity, we can also characterize legality of a finite sequence $a_0 \cdots a_k$ of actions, in terms of satisfiability of the QLTL formula

$$Occurs(a_0 \cdots a_k, rs) \ \wedge$$
$$\forall now.\Box(now \wedge \Diamond\bigcirc rs) \rightarrow LegalNext(\Gamma, \vec{a}, now)$$

5 PLANNING

The logic LTL can express very sophisticated dynamic properties, which can be either verified wrt the system specification, but can also be used as temporally extended goals [1] to synthesize plans. For example, *reachability of a desired state of affairs*, i.e., "is a situation where a given goal φ_{goal} holds reachable?", can be expressed by the formula $\Diamond\varphi_{goal}$. Goals can also be more sophisticated. For example, an *achieving and maintenance goal*, i.e., "is there a sequence of actions

that achieves a certain goal φ_{agoal} while another goal φ_{mgoal} is kept satisfied?", can be expressed by the formula $\varphi_{mgoal}\, \mathcal{U}\, \varphi_{agoal}$. Similarly, *safety*, *invariance*, *liveness*, and *fairness properties* can be expressed in LTL.

We provide a method for *conformant planning* [33] in the setting where both the dynamic system and the goal are specified by arbitrary formulas of LTL. The problem of conformant planning consists in constructing a plan, i.e., a sequence of actions, that guarantees satisfaction of the goal whenever the conditions specified for the system are satisfied. In general, a plan that satisfies an LTL formula needs to be infinite, although it is finitely representable. If the goal can be fulfilled in a finite number of steps and we are interested in a finite plan, we may use a dummy action with no effects, and require it to be executed just after the goal is fulfilled and not before (this can be done by suitably changing the goal). The dummy action acts as a marker for the end of the plan.

Formally, conformant planning can be described as follows: find an (infinite) sequence of actions $a_0 a_1 \cdots$ such that for any sequence $\sigma_0 \sigma_1 \cdots$ of truth assignments to fluents, such that $a_0 a_1 \cdots$ and $\sigma_0 \sigma_1 \cdots$ satisfy the system specification Γ, the goal γ is also satisfied.

Again, this condition cannot be expressed in LTL. However, it can be expressed in QLTL as follows. Let us introduce the formula

$$Plan(\Gamma, \gamma, \vec{a}) \doteq \forall \vec{f}.\Gamma(\vec{a}, \vec{f}) \rightarrow \gamma(\vec{a}, \vec{f})$$

which characterizes the sequences of actions such that, for all interpretations of the fluents consistent with the system specification, the goal is fulfilled. Hence, without considering legality, we can express plan existence as satisfiability of

$$Plan(\Gamma, \gamma, \vec{a})$$

This is a Π_1^{QLTL} formula. Considering that conformant planning is already EXPSPACE-hard for much simpler settings than the one considered here [18, 11] (where each action is always possible), we get:

Theorem 9 *Verifying existence of a (non necessarily legal) plan in* LTL *is EXPSPACE-complete.*

To verify existence of a legal plan, we simply have to check the satisfiability of

$$Plan(\Gamma, \gamma, \vec{a}) \wedge Legal(\Gamma, \vec{a})$$

Observe that, due to the presence of the legality check, this is a Π_2^{QLTL} formula. Hence, by Theorems 4 and 8, we get:

Theorem 10 *Verifying existence of a legal plan in* LTL *is 2-EXPSPACE-complete.*

To actually find a plan (either with legality check or not), one can develop an algorithm based on Büchi automata. Let us consider the case without legality check. Let $\varphi(\vec{a}, \vec{f})$ be $\Gamma(\vec{a}, \vec{f}) \rightarrow \gamma(\vec{a}, \vec{f})$. We have to check satisfiability of $\exists \vec{a}.\forall \vec{f}.\varphi(\vec{a}, \vec{f})$, i.e., $\exists \vec{a}.\neg\exists \vec{f}.\neg\varphi(\vec{a}, \vec{f})$ This can be reduced to checking nonemptiness of a Büchi automaton constructed as follows. We build an automaton A_1 corresponding to the LTL formula $\neg\varphi(\vec{a}, \vec{f})$ (exponential step). Then we project out $\vec{f}$ from such an automaton, obtaining an automaton A_2 accepting sequences of actions (polynomial step). Next we complement A_2 obtaining A_3 (exponential step), and check for nonemptiness of A_3. As mentioned, the nonemptiness algorithm for Büchi automata returns a witness for nonemptiness, which can be interpreted as a plan. The plan returned consists of two parts: a sequence arriving to a certain state, and a second sequence that forms a cycle back into that state. Thus, such plans have finite representations. The construction of the automaton is double exponential in both the size of the system specification Γ and of the goal specification γ. Observe that this algorithm is optimal, wrt complexity, for plan existence, since one can perform complementation and the final intersection on the fly, while checking for nonemptiness [40]. Using the same kind of automata manipulations we can build an automaton for checking existence of plans guaranteed to be legal.

6 CONCLUSIONS

In this paper, we have studied reasoning about action effects, legality of actions, and conformant planning in the setting of LTL. We have provided reasoning techniques based on a second-order extension of LTL.

The results obtained extend immediately to other variants of LTL such as μLTL, which is an extension of LTL with explicit fixpoint operators that is able to express any ω-regular property [37]. The key is that the reasoning techniques rely on an exponential translation of LTL to Büchi automata [39]. Such a translation is also known for μLTL. In addition, one may directly specify the system by a Büchi automaton instead of an LTL formula. Then, adopting the techniques discussed in this paper, it can be shown that reasoning on action effects remains PSPACE-complete, synthesizing non-necessarily legal plans remains EXPSPACE-complete, while testing legality becomes EXPSPACE-complete.

We have adopted a very general non-prescriptive notion of legality of actions, which turned out to be quite

sophisticated and complex to verify. It would be very interesting to study such a notion in the branching-time setting at the base of situation calculus, dynamic logics, branching-time logics, etc. Note that the complexity is probably going to increase, by moving to the branching-time setting, which is in fact richer than the linear-time setting considered here. Also, one should stress that for system specifications of a special form, checking legality of sequences of actions may become much easier. For example, if our system only contains formulas expressing successor state axioms (see Section 3) then all actions are always legal.

Finally, algorithms for checking nonemptiness of Büchi automata, which are at the base of reasoning procedures for LTL and QLTL, have proved to be well suited for scaling up to very large systems. A breakthrough technology has been the use of *symbolic methods* and the experimental results on adopting symbolic techniques for planning under incomplete information are quite promising [7, 8, 5]. The symbolic techniques can be adapted to our framework. So, in spite of the high worst-case complexity, the scalability of the algorithms involved, hint that the automata-theoretic approach may actually be feasible, even in the general setting considered here.

Acknowledgements

The third author was supported in part by NSF grants CCR-9700061, CCR-9988322, IIS-9908435, IIS-9978135, and EIA-0086264, by BSF grant 9800096, and by a grant from the Intel Corporation.

References

[1] F. Bacchus and F. Kabanza. Planning for temporally extended goals. *Ann. of Mathematics and Artificial Intelligence*, 22:5–27, 1998.

[2] F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116(1–2):123–191, 2000.

[3] C. Baral, M. Gelfond, and A. Provetti. Representing actions: laws, observations and hypotheses. *J. of Logic Programming*, 31(1–3):201–243, 1997.

[4] H. Barringer *et al.* MetateM: an introduction. *Formal Aspects of Computing*, 7(5):533–549, 1995.

[5] P. Bertoli, A. Cimatti, and M. Roveri. Heuristic search + symbolic model checking = efficient conformant planning. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pp. 467–472, 2001.

[6] S. Cerrito and M. C. Mayer. Bounded model search in linear temporal logic and its application to planning. In *Proc. of the 2nd Int. Conf. on Analytic Tableaux and Related Methods (TABLEAUX'98)*, *LNAI* 1397, pp. 124–140. Springer, 1998.

[7] A. Cimatti, E. Giunchiglia, F. Giunchiglia, and P. Traverso. Planning via model checking. In *Proc. of the 4th Eur. Conf. on Planning (ECP'97)*, 1997.

[8] A. Cimatti and M. Roveri. Conformant planning via symbolic model checking. *J. of Artificial Intelligence Research*, 13:305–338, 2000.

[9] M. Daniele, P. Traverso, and M. Y. Vardi. Strong cyclic planning revisited. In *Proc. of the 5th Eur. Conf. on Planning (ECP'99)*, *LNAI* 1809, pp. 35–48. Springer, 1999.

[10] G. De Giacomo, L. Iocchi, D. Nardi, and R. Rosati. A theory and implementation of cognitive mobile robots. *J. of Logic and Computation*, 9(5):759–785, 1999.

[11] G. De Giacomo and M. Y. Vardi. Automata-theoretic approach to planning for temporally extended goals. In *Proc. of the 5th Eur. Conf. on Planning (ECP'99)*, *LNAI* 1809, pp. 226–238. Springer, 1999.

[12] P. Doherty, J. Gustafsson, L. Karlsson, and J. Kvarnström. (TAL) Temporal Action Logics: language specification and tutorial. *Elect. Trans. on Artificial Intelligence*, 2(3–4):273–306, 1998.

[13] E. A. Emerson. Automated temporal reasoning about reactive systems. In *Logics for Concurrency: Structure versus Automata*, *LNCS* 1043, pp. 41–101. Springer, 1996.

[14] O. Etzioni *et al.* An approach to planning with incomplete information. In *Proc. of the 3rd Int. Conf. on Knowledge Representation and Reasoning (KR'92)*, pp. 115–125, 1992.

[15] B. Finkbeiner. Language containment checking with nondeterministic BDDs. In *Proc. of the 7th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2001)*, *LNCS* 2031, pp. 24–38. Springer, 2001.

[16] M. Gelfond and V. Lifschitz. Action languages. *Elect. Trans. on Artificial Intelligence*, 3(16), 1998.

[17] M. R. Genesereth and I. R. Nourbakhsh. Time-saving tips for problem solving with incomplete information. In *Proc. of the 11th Nat. Conf. on Artificial Intelligence (AAAI'93)*, pp. 724–730, 1993.

[18] P. Haslum and P. Jonsson. Some results on the complexity of planning with incomplete information. In *Proc. of the 5th Eur. Conf. on Planning (ECP'99)*, 1999.

[19] O. Kupferman and M. Y. Vardi. Weak alternating automata are not that weak. *ACM Trans. on Computational Logic*, 2(3):408–429, 2001.

[20] R. P. Kurshan. *Computer Aided Verification of Coordinating Processes.* Princeton Univ. Press, 1994.

[21] J. Kvarnström and P. Doherty. TALplanner: a temporal logic based forward chaining planner. *Ann. of Mathematics and Artificial Intelligence*, 30:119–169, 2001.

[22] N. McCain and H. Turner. Satisfiability planning with causal theories. In *Proc. of the 6th Int. Conf. on Knowledge Representation and Reasoning (KR'98)*, pp. 212–223, 1998.

[23] J. McCarthy. Circumscription — a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39,171–172, 1980.

[24] R. Miller and M. Shanahan. The event calculus in classical logic — Alternative axiomatisations. *Elect. Trans. on Artificial Intelligence*, 4(16), 1999.

[25] M. Pistore and P. Traverso. Planning as model checking for extended goals in non-deterministic domains. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pp. 479–484, 2001.

[26] A. Pnueli. The temporal logic of programs. In *Proc. of the 18th Annual Symp. on the Foundations of Computer Science (FOCS'77)*, pp. 46–57, 1977.

[27] R. Reiter. Narratives as programs. In *Proc. of the 7th Int. Conf. on Knowledge Representation and Reasoning (KR 2000)*, pp. 99–108, 2000.

[28] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems.* The MIT Press, 2001.

[29] E. Sandewall. *Features and Fluents: The Representation of Knowledge about Dynamical Systems.* Clarendon Press, Oxford, 1994.

[30] M. Shanahan. *Solving the Frame Problem: A Mathematical Investigation of the Common Law of Inertia.* The MIT Press, 1997.

[31] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logic. *J. of the ACM*, 32(3):733–749, 1985.

[32] A. P. Sistla, M. Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987.

[33] D. E. Smith and D. S. Weld. Conformant Graphplan. In *Proc. of the 15th Nat. Conf. on Artificial Intelligence (AAAI'98)*, pp. 889–896, 1998.

[34] M. Thielscher. Causality and the qualification problem. In *Proc. of the 5th Int. Conf. on Knowledge Representation and Reasoning (KR'96)*, pp. 51–62, 1996.

[35] M. Thielscher. Introduction to the fluent calculus. *Elect. Trans. on Artificial Intelligence*, 3(14), 1998.

[36] W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science*, vol. B, pp. 133–192. Elsevier Science, 1990.

[37] M. Y. Vardi. A temporal fixpoint calculus. In *Proc. of the 15th ACM SIGACT-SIGPLAN Symp. on Principles of Programming Languages (POPL'88)*, pp. 250–259, 1988.

[38] M. Y. Vardi. An automata-theoretic approach to fair realizability and synthesis. In *Proc. of the 7th Int. Conf. on Computer Aided Verification (CAV'95)*, *LNCS* 939, pp. 267–292. Springer, 1995.

[39] M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency: Structure versus Automata*, *LNCS* 1043, pp. 238–266. Springer, 1996.

[40] M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.

[41] D. S. Weld and O. Etzioni. The first law of robotics (a call to arms). In *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI'94)*, pp. 1042–1047, 1994.

On the Semantics of Deliberation in IndiGolog — From Theory to Implementation

Giuseppe De Giacomo
Dip. Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
degiacomo@dis.uniroma1.it

Yves Lespérance
Dept. of Computer Science
York University
Toronto, ON, M3J 1P3, Canada
lesperan@cs.yorku.ca

Hector J. Levesque and Sebastian Sardiña
Dept. of Computer Science
University of Toronto
Toronto, ON, M5S 3G4, Canada
{hector,ssardina}@ai.toronto.edu

Abstract

In this paper, we develop an account of the kind of deliberation that an agent that is doing planning or executing high-level programs under incomplete information must be able to perform. The deliberator's job is to produce a kind of plan that does not itself require deliberation to interpret. We characterize these as *epistemically feasible programs*: programs for which the executing agent, at every stage of execution, by virtue of what it knew initially and the subsequent readings of its sensors, always knows what step to take next towards the goal of completing the entire program. We formalize this notion and characterize deliberation in the IndiGolog agent language in terms of it. We also show that for certain classes of problems, which correspond to conformant planning and conditional planning, the search for epistemically feasible programs can be limited to programs of a simple syntactic form. We also discuss implementation issues and execution monitoring and replanning.

1 INTRODUCTION

While a large amount of work on planning deals with issues of efficiency, a number of representational questions remain. This is especially true in applications where because of limitations on the information available at plan time, and quite apart from computational concerns, no *straight-line* plan (that is, no linear sequence of actions) can be demonstrated to achieve a goal. In very many cases, it is necessary to supplement what is known at plan time by information that can only be obtained at run time via sensing.

In cases like these, what should we expect a planner to do given a goal? We cannot expect it to return a straight-line plan. We could get it to return a more general *program* of some sort, but we need to be careful: if the program is general enough, it may be as challenging to figure out how to execute it as it was to achieve the goal in the first place.

This is certainly true for programs in the Golog family of high-level programming languages [Levesque et al., 1997, De Giacomo et al., 2000, Reiter, 2001a]. Those logic languages offer an interesting alternative to planning in which the user specifies not just a goal, but also constraints on how it is to be achieved, perhaps leaving small sub-tasks to be handled by an automatic planner. In that way, a high-level program serves as a "guide" heavily restricting the search space. By a high-level program, we mean one whose primitive instructions are domain-dependent actions of the robot, whose tests involve domain-dependent fluents affected by these actions, and whose code may contain nondeterministic choice points. Instead of looking for a legal sequence of actions achieving some goal, the (planning) task now is to find a sequence that constitutes a legal execution of a high-level program.

At its most basic, planning should be a form of deliberation, whose purpose is to produce a specification of the desired behavior, a specification which should not itself require deliberation to interpret. In [Levesque, 1996] it was suggested that a planner's job was to return a *robot program*, a syntactically-defined structure that a robot could follow while consulting its sensors to determine a conditional course of action. Other forms of conditional plans have been proposed, for example, in [Peot and Smith, 1992, Smith et al., 1998, Lakemeyer, 1999]. What these all have in common, is that they define plans as *syntactically restricted* programs.

In this paper, we consider a different and more abstract version of plans. We propose to treat plans as *epistemically feasible* programs: programs for which the executing agent, at every stage of execution, by virtue of what it knew initially and the subsequent readings of its sensors, always *knows* what step to take next towards the goal of completing the entire program.

This paper will not present algorithms for generating epistemically feasible programs. What we will do, however, is characterize the notion formally, prove that certain cases of syntactically restricted programs are epistemically feasible, and that in some cases where there is an epistemically feasible program, a syntactically restricted one that has the same outcome can also be derived.

To make these concepts precise, it is useful to consider a framework where we can talk about the planning and execution of very general agent programs involving sensing and acting. IndiGolog [De Giacomo and Levesque, 1999a] is a variant of Golog intended to be executed online in an incremental way. Because of this incremental style execution, an agent program is capable of gathering new information from the world during its execution. Most relevant for our purposes is that IndiGolog includes a *search* operator which allows it to only take a step if it can convince itself that the step will allow it to eventually complete some user-specified subprogram. In that way, IndiGolog provides an attractive integrated account of sensing, planning, and action. However, IndiGolog search does not guarantee that it will not get stuck in a situation where it knows that some step can be performed, but does not know which. It is this search operator that we will generalize here.

The rest of the paper is organized as follows. First, in Section 2 we set the stage by presenting the situation calculus and high-level programs based on it. In Section 3, since we are going to make a specific use of the knowledge operator for characterizing the program returned by the deliberator, we introduce *epistemically accurate theories* and a basic property they have w.r.t. reasoning. In Section 4, we characterize *epistemically feasible deterministic programs*, i.e., the kind of program that we consider suitable results of the deliberation process, and in Section 5, we study two notable subclasses of epistemically feasible deterministic programs, that can be characterized in terms of syntax only. In Section 6 we discuss how some of the abstract notions we have introduced can be readily implemented in practice. In Section 7, we discuss how the deliberated program could be monitored and revised if circumstances require it. Finally, in Section 8, we draw conclusions and discuss future and related work.

2 THE SITUATION CALCULUS AND INDIGOLOG

The technical machinery we use to define program execution in the presence of sensing is based on that of [De Giacomo and Levesque, 1999a, De Giacomo et al., 2000]. The starting point in the definition is the situation calculus [McCarthy and Hayes, 1979]. We will not go over the language here except to note the following components: there is a special constant S_0 used to denote the *initial situation*, namely that situation in which no actions have yet occurred; there is a distinguished binary function symbol do where $do(a, s)$ denotes the successor situation to s resulting from performing the action a; relations whose truth values vary from situation to situation, are called (relational) *fluents*, and are denoted by predicate symbols taking a situation term as their last argument; and there is a special predicate $Poss(a, s)$ used to state that action a is executable in situation s. To deal with knowledge and sensing, we follow [Moore, 1985, Scherl and Levesque, 1993, Levesque, 1996] and use a fluent $K(s', s)$ used to represent what situations s' are considered epistemically possible by the agent in situation s. $\mathbf{Know}(\phi(now), s)$ is then taken to be an abbreviation for the formula $\forall s'.K(s', s) \supset \phi(now/s')$. In this paper, we only deal explicitly with sensing actions with binary outcomes as in [Levesque, 1996]. However, the results presented here can be easily generalized to sensors with multiple outcomes. To represent the information provided by a sensing action, we use a predicate $SF(a, s)$, which holds if action a returns the binary sensing result 1 in situation s. For a sensing action $sense_\phi$ that senses the truth value of ϕ, we would have $[SF(sense_\phi, s) \equiv \phi(s)]$, and for any ordinary action a that does not involve sensing, we would use $[SF(a, s) \equiv True]$.

Within this language, we can formulate domain theories which describe how the world changes as the result of the available actions. One possibility is an action theory of the following form [Reiter, 1991, 2001a]:

- Axioms describing the initial situation, S_0.
- Action precondition axioms, one for each primitive action a, characterizing $Poss(a, s)$.
- Successor state axioms, one for each fluent F, stating under what conditions $F(\vec{x}, do(a, s))$ holds as a function of what holds in situation s; these take the place of effect axioms, but also provide a solution to the frame problem.
- Sensed fluent axioms, one for each primitive action a of the form $SF(a, s) \equiv \phi_a(s)$, characterizing SF [Levesque, 1996].
- The following successor state axiom for the knowledge fluent K [Scherl and Levesque, 1993]:

$$K(s'', do(a, s)) \equiv \exists s'.s' = do(a, s') \wedge K(s', s) \wedge Poss(a, s') \wedge [SF(a, s') \equiv SF(a, s)]$$

- Unique names axioms for the primitive actions.
- Some foundational, domain independent axioms [Lakemeyer and Levesque, 1998, Reiter, 2001a].

To describe a run which includes both actions and their sensing results, we use the notion of a *history*, i.e., a sequence of pairs (a, x) where a is a primitive action and x is 1 or 0, a sensing result. Intuitively, the history $(a_1, x_1) \cdot \ldots \cdot (a_n, x_n)$ is one where actions $a_1, \ldots, a_n$ happen starting in some initial situation, and each action a_i returns sensing value x_i. We assume that if a_i is an ordinary action with no sensing, then $x_i = 1$. Notice that the empty sequence ϵ is a history.

We use $end[\sigma]$ as an abbreviation for the situation term called the *end situation* of history σ on the initial situation S_0, and defined by: $end[\epsilon] = S_0$; and inductively, $end[\sigma \cdot (a, x)] = do(a, end[\sigma])$.

We also use $Sensed[\sigma]$ as an abbreviation for a formula of the situation calculus, the *sensing results* of a history, and defined by: $Sensed[\epsilon] = True$; and inductively, $Sensed[\sigma \cdot (a, 1)] = Sensed[\sigma] \wedge SF(a, end[\sigma])$, and $Sensed[\sigma \cdot (a, 0)] = Sensed[\sigma] \wedge \neg SF(a, end[\sigma])$. This formula uses SF to tell us what must be true for the sensing to come out as specified by σ starting in S_0.

Next we turn to programs. The programs we consider here are based on the ConGolog language defined in [De Giacomo et al., 2000], which provides a rich set of programming constructs summarized below:

α,	primitive action
$\phi?$,	wait for a condition
$\delta_1; \delta_2$,	sequence
$\delta_1 \mid \delta_2$,	nondeterministic branch
$\pi x. \delta$,	nondeterministic choice of argument
δ^*,	nondeterministic iteration
if ϕ **then** δ_1 **else** δ_2 **endIf**,	conditional
while ϕ **do** δ **endWhile**,	while loop
$\delta_1 \parallel \delta_2$,	concurrency with equal priority
$\delta_1 \gg \delta_2$,	concurrency with δ_1 at a higher priority
$\delta^{\|}$,	concurrent iteration
$\langle\, \vec{x} : \phi \rightarrow \delta \,\rangle$,	interrupt
$p(\vec{\theta})$,	procedure call[1]

Among these constructs, we notice the presence of of nondeterministic constructs. These include $(\delta_1 \mid \delta_2)$, which nondeterministically chooses between programs δ_1 and δ_2, $\pi x. \delta$, which nondeterministically picks a binding for the variable x and performs the program δ for this binding of x, and δ^*, which performs δ zero or more times. Also notice that ConGolog includes constructs for dealing with concurrency. In particular $(\delta_1 \parallel \delta_2)$ expresses the concurrent execution (interpreted as interleaving) of the programs δ_1 and δ_2. Beside $(\delta_1 \parallel \delta_2)$ ConGolog includes other constructs for dealing with concurrency, such as prioritized concurrency $(\delta_1 \gg \delta_2)$, and interrupts $\langle\, \vec{x} : \phi \rightarrow \delta \,\rangle$. We refer the reader to [De Giacomo et al., 2000] for a detailed account of ConGolog.

In [De Giacomo et al., 2000], a single step transition semantics in the style of [Plotkin, 1981] is defined for ConGolog programs. Two special predicates $Trans$ and $Final$ are introduced. $Trans(p, s, p', s')$ means that by executing program p starting in situation s, one can get to situation s' in one elementary step with the program p' remaining to be executed, that is, there is a possible transition from the configuration (p, s) to the configuration (p', s'). $Final(p, s)$ means that program p may successfully terminate in situation s, i.e., the configuration (p, s) is final.[2]

Offline executions of programs, which are the kind of executions originally proposed for Golog and ConGolog [Levesque et al., 1997, De Giacomo et al., 2000], are characterized using the $Do(p, s, s')$ predicate, which means that there is an execution of program p that starts in situation s and terminates in situation s':

$$Do(p, s, s') \stackrel{\text{def}}{=} \exists p'. Trans^*(p, s, p', s') \wedge Final(p', s'),$$

where $Trans^*$ is the reflexive transitive closure of $Trans$. An offline execution of program p from situation s is a sequence of actions $a_1, \ldots, a_n$ such that:

$$Axioms \models Do(p, s, do(a_n, \ldots, do(a_1, s))).$$

Observe that an offline executor is in fact similar to a planner that given a program, a starting situation, and a theory describing the domain, produces a sequence of action to execute in the environment. In doing this, it has no access to sensing results, which will only be available at runtime. See [De Giacomo et al., 2000] for more details.

In [De Giacomo and Levesque, 1999a], IndiGolog, an extension of ConGolog that deals with online executions with sensing is developed. The semantics defines an *online execution* of an IndiGolog program p starting from a history σ, as a sequence of *online configurations* $(p_0 = p, \sigma_0 = \sigma), \ldots, (p_n, \sigma_n)$ such that for $i = 0, \ldots, n-1$:

$$Axioms \cup \{Sensed[\sigma_i]\} \models Trans(p_i, end[\sigma_i], p_{i+1}, end[\sigma_{i+1}]),$$

$$\sigma_{i+1} = \begin{cases} \sigma_i & \text{if } end[\sigma_{i+1}] = end[\sigma_i], \\ \sigma_i \cdot (a, x) & \text{if } end[\sigma_{i+1}] = do(a, end[\sigma_i]) \\ & \text{and } a \text{ returns } x. \end{cases}$$

[1] For the sake of simplicity, we will not consider procedures in this paper.

[2] For example, the transition requirements for sequence are

$$Trans([p_1; p_2], s, p', s') \equiv Final(p_1, s) \wedge Trans(p_2, s, p', s') \vee \exists q'. Trans(p_1, s, q', s') \wedge p' = (q'; p_2)$$

i.e., to single-step the program $(p_1; p_2)$, either p_1 terminates and we single-step p_2, or we single-step p_1 leaving some q', and $(q'; p_2)$ is what is left of the sequence.

An *online execution successfully terminates* if

$$Axioms \cup \{Sensed[\sigma_n]\} \models Final(p_n, end[\sigma_n]).$$

There is no automatic lookahead in IndiGolog. Instead, a *search* operator $\Sigma(p)$ is introduced to allow the programmer to specify when lookahead should be performed. $Final$ and $Trans$ are defined for the new operator as follows. For $Final$, we simply have that $(\Sigma(p), s)$ is a final configuration of the program if (p, s) itself is, i.e.,

$$Final(\Sigma(p), s) \equiv Final(p, s).$$

For $Trans$, we have that the configuration $(\Sigma(p), s)$ can evolve to $(\Sigma(q'), s')$ provided that (p, s) can evolve to (q', s') and from (q', s') it is possible to reach a final configuration in a finite number of transitions, i.e.,

$$Trans(\Sigma(p), s, p', s') \equiv \exists q', s_f.\ p' = \Sigma(q') \wedge Trans(p, s, q', s') \wedge Do(q', s', s_f).$$

This semantics means that $Axioms \cup \{Sensed[\sigma]\} \models Trans(\Sigma(p), end[\sigma], \Sigma(p'), s')$ iff $Axioms \cup \{Sensed[\sigma]\} \models Trans(p, end[\sigma], p', s')$ and $Axioms \cup \{Sensed[\sigma]\} \models \exists s_f.Do(p', s', s_f)$. Thus, with this definition, the axioms entail that a step of the program can be performed provided that they entail that this step can be extended into a complete execution (i.e., in all models). This prunes executions that are bound to fail later on. But it does not guarantee that the executor will not get stuck in a situation where it knows that some transition can be performed, but does not know which. For example, consider the program $(a; \textbf{if } \phi \textbf{ then } b \textbf{ else } c) \mid d$, where actions a, b, c, and d are always possible, but where the agent does not know whether ϕ holds after a. There are two possible first steps, d which terminates successfully, and a after which the executor is stuck. Unfortunately, Σ does not distinguish between the two cases, since even in the latter, there does exist an (unknown) transition to a final state.

3 EPISTEMICALLY ACCURATE THEORIES

In this paper we are going to look at theories that are *epistemically accurate*, meaning that what is known accurately reflects what the theory says about the dynamic system.[3] Formally, epistemically accurate theories are theories as introduced earlier, but with two additional constraints:

- The initial situation is characterized by an axiom of the form $\textbf{Know}(\phi, S_0)$ where ϕ is an *objective formula*, i.e., a formula where the knowledge fluent K does not appear, which describes the initial situation, S_0. Note that there can be fluents about which nothing is known in the initial situation.
- There is an axiom stating that the accessibility relation K is reflexive in the initial situation, which is then propagated to all situations by the successor state axiom for K [Scherl and Levesque, 1993].

[3] In [Reiter, 2001b] a similar notion is used to deal with knowledge-based programs and reduce knowledge to provability.

For epistemically accurate theories we have established the following result:

Theorem 1 *For any objective sentence about situation s, $\phi(s)$ ($Trans$ and $Final$ may appear in $\phi(s)$), $Axioms \cup \{Sensed[\sigma]\} \models \phi(end[\sigma])$ if and only if $Axioms \cup \{Sensed[\sigma]\} \models \textbf{Know}(\phi, end[\sigma])$.*

Proof Sketch: $\Leftarrow$ Follows trivially from the reflexivity of K in the initial situation, and the fact that it is preserved by the successor state axiom for K.

$\Rightarrow$ Suppose the thesis does not hold, i.e., there exists a model M of $Axioms \cup \{Sensed[\sigma]\}$ such that for some s', $M \models K(s', end[\sigma])$ and $M \models \neg\phi(s')$.

Then take the structure M' obtained from M by intersecting the objects of sort situation with those that are in the situation tree rooted in the initial ancestor situation of s', say s'_0. M' satisfies all the axioms in $Axioms$ except the reflexivity and the successor state axiom for K, and the initial state axiom, which is of the form $\textbf{Know}(\Psi(now), S_0)$ (the other axioms involve neither K nor S_0). Observe that $Trans$ and $Final$ for the situations in the tree are defined by considering relations involving only situations in the same tree.

Now consider the M'' obtained from M' by adding the constant S_0 and making it denote s'_0. Although M' and M'' do not satisfy $\textbf{Know}(\Psi(now), S_0)$, we have that $M'' \models \Psi(S_0)$. Moreover, the successor state axiom for K implies

$$Axioms \cup \{Sensed[\sigma'] \cdot (a, 1)\} \models \textbf{Know}(SF(a, now), end[\sigma' \cdot (a, 1)])$$
$$Axioms \cup \{Sensed[\sigma'] \cdot (a, 0)\} \models \textbf{Know}(\neg SF(a, now), end[\sigma' \cdot (a, 0)])$$

and the fact that the successor state axiom for K holds in M ensures that all predecessors of s' are K accessible from predecessors of $end[\sigma]$ in M. These imply that $M'' \models Sensed[\sigma]$.

Finally let us define M''' by adding to M'' the predicate K and making it denote the identity relation on situations. Then $M''' \models Axioms \cup \{Sensed[\sigma]\}$. On the other hand since $M' \models \neg\phi(s')$, so does M''', a contradiction. ■

This means that if some objective property of the system is entailed, then it is also known and vice-versa.

4 DELIBERATION PROGRAM STEPS

We are going to introduce and semantically characterize the deliberation steps in the program. The basic idea of the semantics we are going to develop is that the task of the deliberator (that performs search) is to try to find a deterministic program that is guaranteed to be "executable" and constitutes a way to execute the program provided, in the sense that it always leads to terminating situations of the given program. Another way to look at this is that the deliberator tries to identify a "strategy" for reaching a final situation of the supplied program. In such a strategy, all choices must be resolved, i.e., the corresponding program needs to be deterministic, and only information that is available to the executor is required. In doing this task, the deliberator performs essentially the same task as the offline executor: it compiles the original program into a simpler program that can be executed without any lookahead. The program it produces however, is not just a linear sequence of actions; it can perform sensing, branching, iteration, etc. Moreover, the program is checked to ensure that the executor will always have enough information to continue the execution. Among other things, this addresses the problem raised above concerning the original semantics of search. Note that our approach is similar to that of [Levesque, 1996]; however, there the strategy was stated in a completely different language (robot programs), here we use ConGolog, i.e., the language used to program the agent itself.

4.1 EPISTEMICALLY FEASIBLE DETERMINISTIC PROGRAMS

The first step in developing this approach is formalizing the notion mentioned above of a deterministic program for which an executor will always have enough information to continue the execution, i.e., will always know what the next step to be performed is. We capture this notion formally by defining the class of *epistemically feasible deterministic programs (EFDPs)* as follows:

$$EFDP(dp, s) \stackrel{\text{def}}{=} \forall dp', s'.Trans^*(dp, s, dp', s') \supset LEFDP(dp', s').$$

$$LEFDP(dp, s) \stackrel{\text{def}}{=} \mathbf{Know}(Final(dp, now), s) \vee \exists dp'.\mathbf{Know}(UTrans(dp, now, dp', now), s) \vee \exists dp', a.\mathbf{Know}(UTrans(dp, now, dp', do(a, now)), s)$$

$$UTrans(dp, s, dp', s') \stackrel{\text{def}}{=} Trans(dp, s, dp', s') \wedge \forall dp'', s''.Trans(dp, s, dp'', s'') \supset dp'' = dp' \wedge s'' = s'$$

Thus to be an $EFDP$, a program must be such that all configurations reachable from the initial program and situation involve a locally epistemically feasible deterministic program ($LEFDP$). A program, is an $LEFDP$ in a situation if the agent knows that it is currently $Final$ or knows what unique transition (with or without an action) it can perform next.

Observe that an epistemically feasible deterministic program is not required to terminate. However, since the agent is guaranteed to know what to do next at every step in its execution, it follows that if it is entailed that the program can reach a final situation, then it can be successfully executed online whatever the sensing outcomes may be:

Theorem 2 *Let dp be such that* $Axioms \cup \{Sensed[\sigma]\} \models EFDP(dp, end[\sigma])$. *Then,* $Axioms \cup \{Sensed[\sigma]\} \models \exists s_f.Do(dp, end[\sigma], s_f)$ *if and only if all online executions of* (dp, σ) *are terminating.*

Proof Sketch: First of all we observe that dp is a deterministic program and its possible online executions from σ are completely determined by the sensing outcomes. We also observe that in each model there will be a single execution of dp, since the sensing outcomes are fully determined in the model.

$\Rightarrow$ If $Axioms \cup \{Sensed[\sigma]\} \models \exists s_f.Do(dp, end[\sigma], s_f)$ then in every model of $Axioms \cup \{Sensed[\sigma]\}$ the only execution of dp from $end[\sigma]$ terminates. Now since offline executions of dp terminate in all models and these models cover all possible sensing outcomes, an online execution must either successfully terminate or get stuck in an online configuration where neither $Final$ nor $Trans$ to some subsequent configuration is entailed. Suppose that there is such an online configuration (dp_i, σ_i) where the agent is stuck. Since in all models of $Axioms \cup \{Sensed[\sigma]\}$ with sensing outcomes as determined by σ_i, $LEFDP(dp_i, end[\sigma_i])$ holds, then either the agent knows that the remaining program is final or knows what the unique next transition is. By reflexivity of K, the agent is correct about this, so $Axioms \cup \{Sensed[\sigma_i]\}$ either entails that dp_i is final or entails that some next transition can be made. If the latter the next transition from (dp_i, σ_i) must be the same in all models of $Axioms \cup \{Sensed[\sigma_i]\}$. Indeed if there were models of $Axioms \cup \{Sensed[\sigma_i]\}$ that had different next transition for $(dp_i, end[\sigma_i])$ then there would be a model where there are distinct epistemic alternatives corresponding to these different models and so the agent would not know what the next transition is in this model. Hence, either way, the agent is not stuck in (dp_i, σ_i), thus getting a contradiction.

$\Leftarrow$ If an online execution of dp from σ terminates it means that the program dp, from $end[\sigma]$, terminates in all models of $Axioms \cup \{Sensed[\sigma]\}$ with the sensing outcomes as in the online execution. Since by hypothesis all online executions terminate, thus covering all possible sensing out-

comes, then dp, from $end[\sigma]$, terminates in all models. ■

4.2 SEMANTICS OF DELIBERATION STEPS

We now give the formal semantics of the deliberation steps. To denote these steps in the program we introduce a *deliberation operator* Δ_e, a new form of the IndiGolog search operator discussed in Section 2.

We define the $Trans$ and $Final$ predicates for the new deliberation operator as follows:

$$Trans(\Delta_e(p), s, dp', s') \equiv$$
$$\exists dp.EFDP(dp, s) \wedge$$
$$\exists s_f.Trans(dp, s, dp', s') \wedge$$
$$Do(dp', s', s_f) \wedge Do(p, s, s_f).$$
$$Final(\Delta_e(p), s) \equiv Final(p, s).$$

Thus, the axioms entail that there is a transition for $\Delta_e(p)$ from a situation s if and only if they entail that there is some epistemically feasible deterministic program dp that reaches a $Final$ situation of the original program p no matter how sensing turns out (i.e., in every model of the axioms). Note also that the remaining program after the transition, dp', is what is left of dp; thus, the agent commits to the strategy/$EFDP$ found in the initial deliberation and executes it.[4] Note that we do not need to put dp' inside a Δ_e block, since it is deterministic.

The following theorem shows that our semantics for the deliberation operator satisfies some basic requirements: if there is a transition for a deliberation block in a history σ, then (1) the program in the deliberation block can reach a $Final$ situation in every model, and (2) so can $\Delta_e(p)$, and moreover (3) $\Delta_e(p)$ can be successfully executed online whatever the sensing results are (thus, the agent will never get to a configuration where it can no longer reach a $Final$ situation or does not know what to do next):

Theorem 3 *If* $Axioms \cup \{Sensed[\sigma]\} \models Trans(\Delta_e(p), end[\sigma], p', s')$, *then*

1. $Axioms \cup \{Sensed[\sigma]\} \models \exists s_f.Do(p, end[\sigma], s_f)$
2. $Axioms \cup \{Sensed[\sigma]\} \models \exists s_f.Do(\Delta_e(p), end[\sigma], s_f)$
3. *All online executions from* $(\Delta_e(p), \sigma)$ *terminate.*

Proof Sketch: 1. and 2. follow immediately from the definition of $Trans$ for Δ_e. For 3. consider that by the definition of $Trans$ for Δ_e, there exists a dp such that $Axioms \cup \{Sensed[\sigma]\} \models EFDP(dp, end[\sigma]) \wedge \exists s_f, p', s'.Trans(dp, end[\sigma], p', s') \wedge Do(p', s', s_f)$. The conditions of Theorem 2 are satisfied, thus we have that all online executions from (dp, σ) are terminating. Since these include all online executions from (p', σ') with $end[\sigma'] = s'$, all online executions from (p', σ') must also be terminating. Hence the thesis follows. ■

[4] We discuss how this commitment to a given "strategy" can be relaxed when we address execution monitoring in Section 7.

5 SYNTAX-BASED ACCOUNTS OF *EFDP*s

In general, deliberating to find a way to execute a high-level program can be very hard because it amounts to doing planning where the class of potential plans is very general. It is thus natural to consider restricted classes of programs. Two particularly interesting such classes are: (i) programs that do not perform sensing, which correspond to conformant plans[5] (see e.g., [Smith and Weld, 1998]), and (ii) programs that are guaranteed to terminate in a bounded number of steps (i.e., do not involve any form of cycles), which correspond to conditional plans (see e.g., [Smith et al., 1998]). We will show that for these two classes, one can restrict one's attention to simple syntactically-defined classes of programs without loss of generality. So if for instance, one is designing a deliberator/planner, one might want to only consider programs from these classes.

5.1 TREE PROGRAMS

Let us now define the class of *(sense-branch) tree programs* $TREE$ with the following BNF rule:

$$dpt ::= nil \mid False? \mid a; dpt_1 \mid True?; dpt_1 \mid sense_\phi; \textbf{if } \phi \textbf{ then } dpt_1 \textbf{ else } dpt_2$$

where a is any non-sensing action, and dpt_1 and dpt_2 are tree programs.

This class includes conditional programs where one can only test a condition that has just been sensed (or trivial tests — these are introduced only for technical reasons). Whenever such a program is executable, it is also epistemically feasible — the agent always knows what to do next:

Theorem 4 *Let dpt be a tree program, i.e.,* $dpt \in TREE$. *Then, for all histories* σ, *if* $Axioms \cup \{Sensed[\sigma]\} \models \exists s_f.Do(dpt, end[\sigma], s_f)$ *then* $Axioms \cup \{Sensed[\sigma]\} \models EFDP(dpt, end[\sigma])$.

Proof Sketch: By induction on the structure of dpt.

Base cases. For nil, it is known that nil is $Final$, so $Axioms \cup \{Sensed[\sigma]\} \models EFDP(nil, end[\sigma])$ holds; for $False?$, the antecedent is false, so the thesis holds.

[5] We remind the reader that conformant plans are sequences of actions that, even under incomplete information about the domain, are guaranteed to reach the desired goal.

Inductive cases. Assume that the thesis holds for dpt_1 and dpt_2. Assume that $Axioms \cup \{Sensed[\sigma]\} \models \exists s_f.Do(dpt, end[\sigma], s_f)$.

For $dpt = a; dpt_1$: $Axioms \cup \{Sensed[\sigma]\} \models \exists s_f.Do(a; dpt_1, end[\sigma], s_f)$ implies that $Axioms \cup \{Sensed[\sigma]\} \models \exists s_f.Do(dpt_1, do(a, end[\sigma]), s_f)$. Since a is a non-sensing action, $Sensed[\sigma \cdot (a,1)] = Sensed[\sigma]$, so we also have that $Axioms \cup Sensed[\sigma \cdot (a,1)]$ entails $\exists s_f.Do(dpt_1, end[\sigma \cdot (a,1)], s_f)$. Thus, by the induction hypothesis, we have $Axioms \cup \{Sensed[\sigma \cdot (a,1)]\} \models EFDP(dpt_1, end[\sigma \cdot (a,1)])$. It follows that $Axioms \cup \{Sensed[\sigma]\} \models EFDP(dpt_1, do(a, end[\sigma]))$. The initial assumption that $Axioms \cup \{Sensed[\sigma]\}$ entails $\exists s_f.Do(a; dpt_1, end[\sigma], s_f)$ also implies that $Axioms \cup \{Sensed[\sigma]\} \models Poss(a, end[\sigma])$ and this must be known by Theorem 1, i.e., $Axioms \cup \{Sensed[\sigma]\} \models$ **Know**$(Poss(a, now), end[\sigma])$. Thus, we have that

$$Axioms \cup \{Sensed[\sigma]\} \models \textbf{Know}(Trans(a; dpt_1, now, dpt_1, do(a, now)), end[\sigma])$$

It is also known that this is the only transition possible for $a; dpt_1$, So $Axioms \cup \{Sensed[\sigma]\} \models LEFDP(a; dpt_1, end[\sigma])$. Therefore,

$$Axioms \cup \{Sensed[\sigma]\} \models EFDP(a; dpt_1, end[\sigma]).$$

For $dpt = True?; dpt_1$: the argument is similar, but simpler since the test does not change the situation.

For $dpt = sense_\phi$; **if** ϕ **then** dpt_1 **else** dpt_2: Suppose that the sensing action returns 1 and let $\sigma_1 = \sigma \cdot (sense_\phi, 1)$. The initial assumption that $Axioms \cup \{Sensed[\sigma]\}$ entails $\exists s_f.Do(dpt, end[\sigma], s_f)$ implies that $Axioms \cup \{Sensed[\sigma_1]\} \models \exists s_f.Do(dpt_1, end[\sigma_1], s_f)$. Thus, by the induction hypothesis, we have $Axioms \cup \{Sensed[\sigma_1)]\} \models EFDP(dpt_1, end[\sigma_1])$. It follows that

$$Axioms \cup \{Sensed[\sigma]\} \models \phi(do(sense_\phi, end[\sigma])) \supset EFDP(dpt_1, do(sense_\phi, end[\sigma])).$$

By a similar argument, it also follows that we must have that

$$Axioms \cup \{Sensed[\sigma]\} \models \neg\phi(do(sense_\phi, end[\sigma])) \supset EFDP(dpt_2, do(sense_\phi, end[\sigma])).$$

The initial assumption $Axioms \cup \{Sensed[\sigma]\} \models \exists s_f.Do(dpt, end[\sigma], s_f)$ also implies that $Axioms \cup \{Sensed[\sigma]\} \models Poss(sense_\phi, end[\sigma])$ and this must be known by Theorem 1, i.e., $Axioms \cup \{Sensed[\sigma]\} \models$ **Know**$(Poss(sense_\phi, now), end[\sigma])$. Thus, we have that

$$Axioms \cup \{Sensed[\sigma]\} \models \textbf{Know}(Trans(dpt, now, \textbf{if } \phi \textbf{ then } dpt_1 \textbf{ else } dpt_2, do(sense_\phi, now)), end[\sigma]).$$

It is also known that this is the only transition possible for dpt, so $Axioms \cup \{Sensed[\sigma]\} \models LEFDP(dpt, end[\sigma])$. Thus, $Axioms \cup \{Sensed[\sigma]\} \models EFDP(dpt, end[\sigma])$. ■

By Theorem 2, we also have that under the conditions of the above theorem, all online executions of (dpt, σ) are terminating. The problem of finding a tree program that yields an execution of a program in a deliberation block is the analogue in our framework of conditional planning (under incomplete information) in the standard setting [Peot and Smith, 1992, Smith et al., 1998].

Next, we show that tree programs are sufficient to express any strategy where there is a known bound on the number of steps it needs to terminate. That is, for any epistemically feasible deterministic program for which this condition holds, there is a tree program that produces the same executions:

Theorem 5 *For any program dp that is*

1. *an epistemically feasible deterministic program, i.e.,* $Axioms \cup \{Sensed[\sigma]\} \models EFDP(dp, end[\sigma])$ *and*
2. *such that there is a known bound on the number of steps it needs to terminate, i.e., where there is an* n *such that*

$$Axioms \cup \{Sensed[\sigma]\} \models \exists p', s', k.k \leq n \wedge Trans^k(dp, end[\sigma], p', s') \wedge Final(p', s')$$

there exists a tree program $dpt \in TREE$ *such that* $Axioms \cup \{Sensed[\sigma]\} \models \forall s_f.Do(dp, end[\sigma], s_f) \equiv Do(dpt, end[\sigma], s_f)$.

Proof Sketch: We construct the tree program $dpt = m(dp, \sigma)$ from dp using the following rules:

- $m(dp, \sigma) = False?$ iff $Axioms \cup \{Sensed[\sigma]\}$ is inconsistent, otherwise
- $m(dp, \sigma) = nil$ iff $Axioms \cup \{Sensed[\sigma]\} \models Final(dp, end[\sigma])$, otherwise
- $m(dp, \sigma) = a; m(dp', \sigma \cdot (a,1))$ iff

$$Axioms \cup \{Sensed[\sigma]\} \models Trans(dp, end[\sigma], dp', do(a, end[\sigma]))$$

for some non-sensing action a,

- $m(dp, \sigma) = sense_\phi$; **if** ϕ **then** $m(dp', \sigma \cdot (sense_\phi, 1))$ **else** $m(dp', \sigma \cdot (sense_\phi, 0))$ iff

$$Axioms \cup \{Sensed[\sigma]\} \models Trans(dp, end[\sigma], dp', do(sense_\phi, end[\sigma]))$$

for some sensing action $sense_\phi$,
- $m(dp, \sigma) = True?; m(dp', \sigma)$ iff

$$Axioms \cup \{Sensed[\sigma]\} \models Trans(dp, end[\sigma], dp', end[\sigma]).$$

Let us show that

$$Axioms \cup \{Sensed[\sigma]\} \models Do(dp, end[\sigma], s_f) \equiv Do(m(dp, \sigma), end[\sigma], s_f).$$

It turns out that, under the hypothesis of the theorem, for all dp and all σ, (dp, σ) is bisimilar to $(m(dp, \sigma), \sigma)$ with respect to online executions. Indeed, it is easy to check that the relation $[(dp, \sigma), (m(dp, \sigma), \sigma)]$ is a bisimulation, i.e., for all dp and σ, $[(dp, \sigma), (m(dp, \sigma), \sigma)]$ implies that

- $Axioms \cup \{Sensed[\sigma]\} \models Final(dp, end[\sigma])$ iff $Axioms \cup \{Sensed[\sigma]\} \models Final(m(dp, \sigma), end[\sigma])$,
- for all dp', σ' if $Axioms \cup \{Sensed[\sigma]\} \models Trans(dp, end[\sigma], dp', end[\sigma'])$ with the set $Axioms \cup \{Sensed[\sigma']\}$ being consistent, then

$$Axioms \cup \{Sensed[\sigma]\} \models Trans(m(dp, \sigma), end[\sigma], m(dp', \sigma'), end[\sigma'])$$

and $[(dp', \sigma'), (m(dp', \sigma'), \sigma')]$,
- for all dp', σ' if $Axioms \cup \{Sensed[\sigma]\} \models Trans(m(dp, \sigma), end[\sigma], m(dp', \sigma'), end[\sigma'])$ with $Axioms \cup \{Sensed[\sigma']\}$ consistent, then

$$Axioms \cup \{Sensed[\sigma]\} \models Trans(dp, end[\sigma], dp', end[\sigma'])$$

and $[(dp', \sigma'), (m(dp', \sigma'), \sigma')]$.

Now, assume that $Axioms \cup \{Sensed[\sigma]\}$ entails $\exists s_f.Do(dp, end[\sigma], s_f)$. Then since dp is an $EFDP$, by Theorem 2 all online execution from (dp, σ) terminate. Hence since (dp, σ) and $(m(dp, \sigma), \sigma)$ are bisimilar, $(m(dp, \sigma), \sigma)$ has the same online executions (apart from the program appearing in the configurations).

Next, observe that given an online execution of (dp, σ) terminating in (dp_f, σ_f), in all models of $Axioms \cup \{Sensed[\sigma]\}$ with sensing outcomes as in σ_f both the program dp and $m(dp, \sigma)$ reach the same situation $end[\sigma_f]$. Since there are terminating online executions for all possible sensing outcomes, the thesis follows. ■

This theorem shows that if we restrict our attention to $EFDP$s that terminate in a bounded number of steps, then we can further restrict our attention to programs of a very specific syntactic form, without any loss in generality. This may simplify the task of coming up with a successful strategy for a given deliberation block.

5.2 LINEAR PROGRAMS

Let the class of linear programs $LINE$ be defined by the following BNF rule:

$$dpl ::= nil \mid a; dpl_1 \mid True?; dpl_1$$

where a is any non-sensing action, and dpl_1 is a linear program.

This class only includes sequences of actions or trivial tests. So whenever such a plan is executable, then it is also epistemically feasible — the agent always knows what to do next:

Theorem 6 *Let dpl be a linear program, i.e., dpl* $\in LINE$. *Then, for all histories* σ, *if* $Axioms \cup \{Sensed[\sigma]\} \models \exists s_f.Do(dpl, end[\sigma], s_f)$ *then* $Axioms \cup \{Sensed[\sigma]\} \models EFDP(dpl, end[\sigma])$.

Proof Sketch: This is a corollary of Theorem 4 for tree programs. Since linear programs are tree programs, the thesis follows immediately from this theorem. ■

By Theorem 2, we also have that under the conditions of the above theorem, all online executions of (dpl, σ) are terminating. Since the agent may have incomplete knowledge, the problem of finding a linear program that yields an execution of a program in a deliberation block is the analogue in our framework of conformant planning in the standard setting [Smith and Weld, 1998].

Next, we show that linear programs are sufficient to express any strategy that does not perform sensing.

Theorem 7 *For any dp that does not include sensing actions, such that* $Axioms \cup \{Sensed[\sigma]\} \models EFDP(dp, end[\sigma])$, *there exists a linear program dpl such that* $Axioms \cup \{Sensed[\sigma]\} \models \forall s_f.Do(dp, end[\sigma], s_f) \equiv Do(dpl, end[\sigma], s_f)$.

Proof Sketch: We show this using the same approach as for Theorem 5 for tree programs. Since dp cannot contain sensing actions, the construction method used in the proof of Theorem 5 produces a tree program that contains no branching and is in fact a linear program. Then, by the same argument as used there, the thesis follows. ■

Observe that this implies that if no sensing is possible — for instance, because there are no sensing actions — then linear programs are sufficient to express every strategy.

Let Δ_l be a deliberation operator that is axiomatized just as Δ_e except that we replace the requirement that dp be an epistemically feasible deterministic program by the requirement that it be a linear program, i.e., where we use the

axiom (the *LINE* predicate is defined in the obvious way):

$$Trans(\Delta_l(p), s, dpl', s') \equiv$$
$$\exists dpl.LINE(dpl) \wedge$$
$$\exists s_f.Trans(dpl, s, dpl', s') \wedge$$
$$Do(dpl', s', s_f) \wedge Do(p, s, s_f).$$

Then, one can show that a program using this deliberation operator $\Delta_l(p)$ can make a transition in a history if and only if one can identify a sequence of actions that is an execution of p in all models for the history:

Theorem 8 *There exists a situation s_f such that*

$$Axioms \cup \{Sensed[\sigma]\} \models Do(p, end[\sigma], s_f)$$

if and only if there is a $dpl \in LINE$ and an s' such that

$$Axioms \cup \{Sensed[\sigma]\} \models Trans(\Delta_l(p), end[\sigma], dpl, s')$$

Proof Sketch: $\Leftarrow$ By hypothesis there exists a dpl that is a *LINE*. If $s' = end[\sigma]$ and then $dpl = true?; dpl'$ and if $s' = do(a, end[\sigma])$, for some action a, and then $dpl = a; dpl'$. In both cases dpl' must be a *LINE*. In every model dpl' reaches from s' a final situation of the original program p. Observe that such a situation will be the same in every model since the sequence of actions starting from s' is fixed by dpl'. It follows that the sequence of action done by dpl starting from s reaches a situation s_f such that $Axioms \cup \{Sensed[\sigma]\} \models Do(p, end[\sigma], s_f)$.

$\Rightarrow$ If for some s_f we have $Axioms \cup \{Sensed[\sigma]\} \models Do(p, end[\sigma], s_f)$ then the sequence of actions from $end[\sigma]$ to s_f is a *LINE* program, which trivially satisfies the left-hand-side of the axiom for Δ_l. Observe that if $s_f = end[\sigma]$ then the linear program can be simply nil. ■

This provides the basis for a simple implementation.

6 IMPLEMENTATION

Let us now examine how the deliberation construct can be implemented according to the specification given above, i.e., by having the interpreter look for an epistemically feasible deterministic program of a certain type, linear, tree, etc. We also relate these implementations to earlier implementation proposals for IndiGolog.

The simplest type of implementation is one that only considers linear programs as potential strategies for executing the program in the deliberation block, as in the specification of Δ_l above. This will work if there is a solution that does not do sensing. Here is the code in Prolog:

```
/* implementation using linear programs  */
trans(delib_l(P),H,DPL1,H1):-
  buildLine(P,DPL,H), trans(DPL,H,DPL1,H1).
buildLine(P,[],H):- final(P,H).
buildLine(P,[(true)?|DPL],H):-
  trans(P,H,P1,H), buildLine(P1,DPL,H).
buildLine(P,[A|DPL],H):-   /* A is not   */
  trans(P,H,P1,[(A,1)|H]), /* a sensing  */
  buildLine(P1,DPL,[(A,1)|H]). /* action */
```

Instead of situations, this code uses histories, which are essentially lists of pairs of actions and sensing outcomes since the initial situation. The `buildLine(P,DPL,H)` predicate basically looks for a sequence of transitions that the program can perform and that that is guaranteed to lead to a final configuration without performing sensing (sensing outcomes for non-sensing actions are assumed to be 1). This approach to implementing deliberation is essentially that used in [De Giacomo et al., 1998, Lespérance and Ng, 2000, De Giacomo et al., 2001], as these assume that deliberation blocks do not contain sensing actions.

A more general type of implementation is one that considers tree programs as potential strategies for executing the program in the deliberation block, assuming that binary sensing actions are available. This can be implemented by generalizing the above as follows:

```
/* implementation using tree programs    */
trans(delib_t(P),H,DPT1,H1):-
   buildTree(P,DPT,H), trans(DPT,H,DPT1,H1).
buildTree(P,[],H):- final(P,H).
buildTree(P,[(true)?|DPT],H):-
  trans(P,H,P1,H), buildTree(P1,DPT,H).
buildTree(P,[A,if(F,DPT1,DPT2)]):-
  trans(P,H,P1,[(A,_)|H]), senses(A,F),
  buildTree(P1,DPT1,[(A,1)|H]),
  buildTree(P1,DPT2,[(A,0)|H]).
buildTree(P,[A|DPT],H):-
  trans(P,H,P1,[(A,_)|H]), not senses(A,_),
  buildTree(P1,DPT,[(A,1)|H]).
buildTree(P,(false)?,H):- inconsistent(H).

inconsistent([(A,1)|H]):- inconsistent(H) ;
    senses(A,F), holds(neg(F),H).
inconsistent([(A,0)|H]):- inconsistent(H) ;
    senses(A,F), holds(F,H).
```

A transition is performed on a program `search_t(p)` only if it is always possible to extend it into a complete execution of `p`. To ensure this, whenever a binary sensing action is encountered, the code verifies the existence of complete executions for both potential sensing outcomes `0` and `1` (3rd clause of `buildTree`). For non-sensing actions, the sensing outcome is assumed to be `1`, and the existence of an execution is verified in this single case (4th clause of `buildTree`). This implementation is similar to that of [De Giacomo and Levesque, 1999a]. Both of the above implementations are sound but not complete.[6]

[6]The incompleteness comes from the fact that they stick to

7 DELIBERATION WITH EXECUTION MONITORING

So far, we have provided a formal account of plans that are suitable for an agent capable of sensing the environment during the execution of a high-level program. We have not addressed, though, another important feature of complex environments with which a realistic agent needs to cope as well: *exogenous actions*. Intuitively, an exogenous action is an action outside the control of the agent, perhaps a natural event or an action performed by another agent. Technically, these are primitive actions that may occur without being part of the user-specified program. It is not hard to imagine how one would slightly alter the definition of *online execution* of Section 2 so as to allow for the occurrence of exogenous actions after each legal transition. Nonetheless, an exogenous action can potentially compromise the online execution of a deliberation block. This is due to the fact that Δ_e commits to a particular EFDP which can turn out to be impossible to execute after the occurrence of some interfering outside action. If there is another EFDP that could be used instead to complete the execution of the deliberation block, we would like the agent to switch to it.

To address this problem, the search operator defined in [Lespérance and Ng, 2000] implements an execution monitoring mechanism. The idea is to recompute a search block whenever the current plan has become invalid due to the occurrence of exogenous actions during the incremental execution. The new search starts from the original program and situation (this is important because often commitments are made early on in the program's execution, and these may have to be revised when an exogenous change occurs) and ensures that the plan produced is compatible with the already performed actions.

Based on [De Giacomo et al., 1998], one can come up with a clean and abstract formalization of execution monitoring and replanning for our epistemic version of deliberation described in Section 4.2. The idea is to avoid permanently committing to a particular EFDP. Instead, we define a deliberation operator Δ_{em} that monitors the execution of the selected EFDP and replans when necessary, possibly selecting an alternative EFDP to follow. The semantics of this *monitored deliberation* construct goes as follows:

$$Trans(\Delta_{em}(p), s, p', s') \equiv$$
$$\exists dp, dp'.EFDP(dp, s) \wedge p' = mnt(dp', s', p, s) \wedge$$
$$\exists s_f.Trans(dp, s, dp', s') \wedge$$
$$Do(dp', s', s_f) \wedge Do(p, s, s_f).$$
$$Final(\Delta_{em}(p), s) \equiv Final(p, s).$$

The main difference is in the remaining program which contains not only the epistemically feasible strategy chosen, but also the original program p, original situation s, and next expected situation s'. These components are packaged using a new language construct mnt, which basically means that the agent should *monitor* the execution of the selected strategy dp using the original program and situation to replan when necessary.

The next step, then, is to define the semantics for the new "monitoring" construct mnt. With that objective, we first introduce two auxiliary relations. Relation $perturbed(mnt(dp, s_e, p_i, s_i), s)$ states whether the strategy dp has just been perturbed in situation s by some exogenous action. There are obviously several ways to define when a strategy has been perturbed. A sensible one is the following: a strategy has been perturbed if the exogenous actions that just occurred rule out a successful execution for both the strategy and the original program of the deliberation block.

$$perturbed(mnt(dp, s_e, p_i, s_i), s) \equiv$$
$$s_e \neq s \wedge \neg\exists s_f.[Do(dp, s, s_f) \wedge Do(p_i \parallel p_{ex}, s_i, s_f)]$$

Notice that we make use of the special program $p_{ex} \stackrel{\text{def}}{=} (\pi a.Exo(a)?; a)^*$, see [De Giacomo et al., 2000], to allow for a legal sequence of exogenous actions. Also, observe that a strategy can be perturbed *only* if an action outside the strategy occurred, in which case the actual situation s would differ from the expected situation s_e. Thus in practice, there is no need to check for perturbation unless an exogenous action or an action other than that performed by the chosen strategy occurs.

The next auxiliary relation is used to calculate a *recovered* strategy dp_r when the current one dp was perturbed in situation s. A sensible definition for it is:

$$recover(mnt(dp, s_e, p_i, s_i), s, dp_r) \equiv$$
$$\exists p_i'.Trans^*(p_i \parallel p_{ex}, s_i, p_i' \parallel p_{ex}, s) \wedge$$
$$EFDP(dp_r, s) \wedge \exists s_f.Do(dp_r, s, s_f) \wedge Do(p_i', s, s_f).$$

Observe that the above definition may end up choosing an *alternative* epistemically feasible strategy than the one chosen before. In a nutshell, a new *recovered* strategy is an epistemically feasible one that is able to "solve" the original program p_i while accounting for *every* action executed

the form of the program while the semantics does not. One example that brings this out is: $\phi?; \psi?; a \mid \neg\phi?; \neg\psi?; a$, where it is known that $\phi \equiv \psi$. For our semantics, the LINE program $True?; True?; a$ is a strategy for executing it, but the implementations fail to find it.

so far, either by the deliberation block or not, since the beginning of the deliberation block.

We now have all the machinery needed to define the semantics for the monitoring construct mnt:

$$\begin{aligned}
&Trans(mnt(dp, s_e, p_i, s_i), s, p', s') \equiv \\
&\quad [\neg perturbed(mnt(dp, s_e, p_i, s_i), s) \wedge \\
&\quad\ \exists dp'.Trans(dp, s, dp', s') \wedge \\
&\quad\quad p' = mnt(dp', s', p_i, s_i)] \vee \\
&\quad [perturbed(mnt(dp, s_e, p_i, s_i), s) \wedge \\
&\quad\ \exists dp_r.recover(mnt(dp, s_e, p_i, s_i), s, dp_r) \wedge \\
&\quad\quad \exists dp'.Trans(dp_r, s, dp', s') \wedge \\
&\quad\quad\quad p' = mnt(dp', s', p_i, s_i)]
\end{aligned}$$

$$\begin{aligned}
&Final(mnt(dp, s_e, p_i, s_i), s) \equiv \\
&\quad [\neg perturbed(mnt(dp, s_e, p_i, s_i), s) \wedge Final(dp, s)] \\
&\quad \vee [perturbed(mnt(dp, s_e, p_i, s_i), s) \wedge \\
&\quad\quad Do(p_i \parallel p_{ex}, s_i, s)]
\end{aligned}$$

For $Trans$, we have two possibilities: (i) if the strategy has not been perturbed, then we continue its execution by performing one step and updating the next expected situation; (ii) if the strategy has just been perturbed, a recovered new strategy dp_r is computed and the execution continues with respect to this alternative strategy. It is important to note that the original program and situation are always kept throughout the whole execution of a deliberation block. In that way, the recovery process can be as general as possible. The case for $Final$ is simpler: (i) if the strategy has not been perturbed, then we check whether the strategy is final in the actual situation; (ii) if the strategy has been perturbed, then there is a chance that the original program might be terminating in the current situation and we check for this.

Summarizing, deliberation can be naturally integrated with execution monitoring in order to cope with exogenous actions that make the chosen strategy unsuitable.

8 CONCLUSION

In this paper, we developed an account of the kind of deliberation that an agent that is doing planning or executing high-level programs must be able to perform. The deliberator's job is to produce a kind of plan that does not itself require deliberation to interpret. We characterized these as *epistemically feasible* programs: programs for which the executing agent, at every stage of execution, by virtue of what it knew initially and the subsequent readings of its sensors, always *knows* what step to take next towards the goal of completing the entire program. We formalized this notion and characterized deliberation in the IndiGolog agent language in terms of it. We have also shown that for certain classes of problems, which correspond to conformant planning and conditional planning, the search for epistemically feasible programs can be limited to programs of a simple syntactic form.

There has been a lot of work in the past on formalizing the notion of epistemically feasible plan, e.g. Moore [1985], Davis [1994], Lespérance et al. [2000], Levesque [1996], and our accounts builds on this. One of its distinguishing features is that it is integrated with the transition system semantics of our programming language. In Lespérance [2001], a similar approach is used to formalize a notion of epistemic feasibility for multiagent system specifications. In McIlraith and Son [2001], a notion of "self-sufficient program" very similar to $EFDP$s is formalized; but this account is more sensitive to the syntax of the program than ours.

In this paper, we have only dealt with binary sensing actions. However, the account of deliberation developed in Section 4 and its extension to provide execution monitoring in Section 7 do not rely on this restriction and apply unchanged to theories with sensing actions that have even an infinite number of possible sensing outcomes.[7] This comes from the fact that our characterization of "good execution strategies" through the notion of $EFDP$ is not syntactic, only requiring the agent to know what action to do next at every step. The results of Section 5.1 showing that tree programs are sufficient to solve any planning/deliberation problem where there is some strategy that solves the problem in a bounded number of steps also generalize to domains involving sensing actions with non-binary but finitely many outcomes; this is easy to see given that any such sensing action can be encoded as a sequence binary sensing actions that read the outcome one bit at a time (one could of course extend the class of tree programs with a non-binary branching structure to avoid the need for such an encoding). Whether a similar characterization can be obtained for sensing actions with an infinite number of possible outcomes is an open problem. While the above holds in principle, as soon as the number of sensing outcomes is more than a few, conditional planning becomes impractical without advice from the programmer as to what conditions the plan should branch on [Lakemeyer, 1999, Thielscher, 2001]. In [Sardiña, 2001], a search construct for IndiGolog that generates conditional plans involving non-binary sensing actions by relying on such programmer advice is developed. This approach seems very compatible with ours and it would be interesting to formalize it as a special case of our account of deliberation. There are also more general theories of sensing, such as that of [De Giacomo and Levesque, 1999b] which deals with online sensors that always provide values and situations where the law of inertia is not always applicable. In [De Giacomo et al., 2001], a search operator for such theories is devel-

[7] One can introduce non-binary sensing actions in our framework as in [Scherl and Levesque, 1993].

oped. It would be worthwhile examining whether this setting could also be handled within our account of deliberation. As well, one could look for syntactic characterizations for certain classes of epistemically feasible deterministic programs in this setting.

References

Ernest Davis. Knowledge preconditions for plans. *Journal of Logic and Computation*, 4(5):721–766, 1994.

Giuseppe De Giacomo, Yves Lespérance, and Hector J. Levesque. ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121:109–169, 2000.

Giuseppe De Giacomo and Hector J. Levesque. An incremental interpreter for high-level programs with sensing. In Hector J. Levesque and Fiora Pirri, editors, *Logical Foundations for Cognitive Agents*, pages 86–102. Springer-Verlag, 1999a.

Giuseppe De Giacomo and Hector J. Levesque. Progression and regression using sensors. In *Proc. of IJCAI-99*, pages 160–165, 1999b.

Giuseppe De Giacomo, Hector J. Levesque, and Sebastian Sardiña. Incremental execution of guarded theories. *ACM Transactions on Computational Logic*, 2(4):495–525, 2001.

Giuseppe De Giacomo, Raymond Reiter, and Mikhail Soutchanski. Execution monitoring of high-level robot programs. In *Proc. of KR-98*, pages 453–465, 1998.

Gerhard Lakemeyer. On sensing and off-line interpreting in Golog. In H. J. Levesque and F. Pirri, editors, *Logical Foundations for Cognitive Agents*, pages 173–187. Springer-Verlag, 1999.

Gerhard Lakemeyer and Hector J. Levesque. AOL: A logic of acting, sensing, knowing, and only-knowing. In *Proc. of KR-98*, pages 316–327, 1998.

Yves Lespérance. On the epistemic feasibility of plans in multiagent systems specifications. In J.-J. Meyer, M. Tambe, and D. Pynadath, editors, *Intelligent Agents VIII, Agent Theories, Architectures, and Languages, 8th Intl. Workshop, ATAL-2001, Seattle, WA, USA, Aug. 1-3, 2001, Proc.*, LNAI. Springer, 2001.

Yves Lespérance, Hector J. Levesque, Fangzhen Lin, and Richard B. Scherl. Ability and knowing how in the situation calculus. *Studia Logica*, 66(1):165–186, 2000.

Yves Lespérance and Ho-Kong Ng. Integrating planning into reactive high-level robot programs. In *Proc. of the Second International Cognitive Robotics Workshop*, pages 49–54, 2000.

Hector J. Levesque. What is planning in the presence of sensing? In *Proc. of AAAI-96*, pages 1139–1146, 1996.

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(59–84), 1997.

John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intellig ence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, 1979.

Sheila McIlraith and Tran Cao Son. Adapting Golog for programming the semantic web. In *Working Notes of the 5th Int. Symposium on Logical Formalizations of Commonsense Reasoning*, pages 195–202, 2001.

Robert C. Moore. A formal theory of knowledge and action. In J. R. Hobbs and Robert C. Moore, editors, *Formal Theories of the Common Sense World*, pages 319–358. Ablex Publishing, Norwood, NJ, 1985.

Mark A. Peot and David E. Smith. Conditional nonlinear planning. In *Proc. of the First International Conference on AI Planning Systems*, pages 189–197, 1992.

Gordon Plotkin. A structural approach to operational semantics. Technical Report DAIMI-FN-19, Computer Science Dept., Aarhus University, Denmark, 1981.

Raymond 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 359–380. Academic Press, 1991.

Raymond Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001a.

Raymond Reiter. On knowledge-based programming with sensing in the situation calculus. *ACM Transactions on Computational Logic*, 2(4):433–457, 2001b.

Sebastian Sardiña. Local conditional high-level robot programs. In *Proc. of LPAR-01*, volume 2250 of *LNAI*, pages 110–124, 2001.

Richard B. Scherl and Hector J. Levesque. The frame problem and knowledge-producing actions. In *Proc. of AAAI-93*, pages 689–695. AAAI Press/The MIT Press, 1993.

David E. Smith, Corin R. Anderson, and Daniel S. Weld. Extending graphplan to handle uncertainty and sensing actions. In *Proc. of AAAI-98*, pages 897–904, 1998.

David E. Smith and Daniel S. Weld. Conformant graphplan. In *Proc. of AAAI-98*, pages 889–896, 1998.

Michael Thielscher. Inferring implicit state knowledge and plans with sensing actions. In *Proc. of KI-01*, volume 2174 of *LNAI*, pages 366–380. Springer, 2001.

ACTIONS AND OTHER EVENTS IN SITUATION CALCULUS

John McCarthy
Computer Science Department
Stanford University
Stanford, CA 94305
jmc@cs.stanford.edu
http://www-formal.stanford.edu/jmc/

Abstract

This article presents a situation calculus formalism featuring events as primary and the usual actions as a special case. Events that are not actions are called *internal events* and actions are called *external events*. The effects of both kinds of events are given by effect axioms of the usual kind. The actions are assumed to be performed by an agent as is usual in situation calculus. An internal event e occurs in situations satisfying an *occurrence assertion* for that event.

A formalism involving actions and internal events describes what happens in the world more naturally than the usual formulations involving only actions supplemented by state constraints. Ours uses only ordinary logic without special causal implications. It also seems to be more *elaboration tolerant*.

The first example is the buzzer with only internal events and which cannot be treated at all with state constraints, because the system never settles down to a steady state.

Our second example is the stuffy room scenario. One occurrence axiom states that when both vents are blocked and the room isn't stuffy, the event $Getstuffy$ occurs. State constraints are unneeded. The stuffy room formalization tolerates an elaboration asserting that when the room becomes stuffy someone unblocks a vent. If we further add that someone else then finds the room cold and blocks the vent again, we get a system that oscillates.

The third example is the blocks world.

The nonmonotonic reasoning involves circumscribing occurrences, changes, and prevention one situation at a time.

Then we offer a general viewpoint on the situation calculus and its applications to real world problems. It relates the formalism of [MH69] which regards a situation as a snapshot of the world to situation calculus theories involving only a few fluents.

1 Introduction: Actions and other events

This article emphasizes the idea that an action by an agent is a particular kind of event. The idea of event is primary and an action is a special case. The treatment is simpler than those regarding events as *natural actions*.

The main features of our treatment are as follows.

1. There are the usual *effect axioms* involving the function $Result(e, s)$, the situation that results when event e occurs in situation s.

2. There are *occurrence axioms* giving conditions for an event to occur. They have the form $conditions(s) \rightarrow occurs(e, s)$.

3. The theory distinguishes between *external events* for which occurrence axioms are not given and *internal events* governed by occurrence axioms. Older treatments of situation calculus often do not provide for internal events. Usually human actions are properly treated as external events, but if the theory contains assertions that a person will perform a certain action, then such an assertion can be given by an occurrence axiom, and the action is an internal event. We include an example of an elaboration of the theory of Ginsberg's stuffy room scenario [GS88] that uses an occurrence axiom to assert that a person will unblock a vent when the room becomes stuffy. Thus an action

can be either an external or internal event. Elaborating the theory by giving an occurrence axiom for the action makes it an internal action in the elaborated theory.

4. Our theories are nonmonotonic and minimize certain predicates situation by situation. The approach is proposed only when information about the future is obtained only by projection from earlier situations. Thus it is not appropriate for the stolen car scenario.

We use internal events instead of state constraints in the stuffy room example. Thus we say that when the vents are blocked, the room becomes stuffy rather than regarding stuffiness as a state constraint. This is closer to human common sense reasoning and natural language usage, as well as being logically simpler.

We begin with formalizing a buzzer which has only internal events, continue with the stuffy room scenario which has both. Our third example is the blocks world.

After these examples, we discuss the nonmonotonic reasoning.

Then we offer a general viewpoint on the situation calculus and its applications to real world problems. It relates the formalism of [MH69] which regards a situation as a snapshot of the world to situation calculus theories involving only a few fluents.

2 The situation calculus formalism

Situations are denoted by the letter s decorated with subscripts. Constants like $S0$ are capitalized, and variables are lower case. However, we do not assume that all situations are generated from a big bang situation $S0$ as does Reiter [Rei01].

The sentence $Holds(pfluent, s)$ asserts that the propositional fluent $pfluent$ holds in the situation s. Sometimes we write just $pfluent(s)$, but the notation with $Holds$ allows quantifying over fluents. We also have term fluents, and $Value(tfluent, s)$ gives the value of $tfluent$ in the situation s. We also sometimes write just $tfluent(s)$.

$Result(e, s)$ denotes the situation that arises when the event e occurs in the situation s. In this simple formalism, neither situations nor events have durations.

$Occurs(e, s)$ is the assertion that the event e occurs in the situation s.

$Next(s)$ is the next situation after s. It is defined by

$$Occurs(e, s) \rightarrow Next(s) = Result(e, s) \qquad (1)$$

for those situations in which an occurrence assertion determines what event will occur.

As an example, three of the axioms of the stuffy room phenomenon are

$$\begin{aligned} &Holds(Blocked1, s) \wedge Holds(Blocked2, s) \\ &\quad \wedge \neg Holds(Stuffy, s) \\ &\quad \rightarrow Occurs(Getstuffy, s), \qquad (2) \\ &Holds(Stuffy, Result(Getstuffy, s)), \quad \text{and} \\ &Holds(Blocked1, Result(Block1, s)). \end{aligned}$$

$Getstuffy$ is an internal event and occurs all by itself when the vents are blocked.

We use circumscription to minimize occurrences, to minimize change (frame problem), and to minimize the fluents that prevent actions and other events (qualification problem).

Treating internal and external events by the same formalism admits elaborations that turn some instances of external events into internal events. Thus we can elaborate the stuffy room scenario by adjoining an occurrence axiom saying that when the room becomes stuffy, someone unblocks a vent, which makes the room unstuffy. The further elaboration that when a vent is unblocked, someone blocks it again, perhaps from feeling cold, causes the system to oscillate and never settle down.

An external event can create a situation in which the occurrence axiom for an internal event is satisfied. This leads to a new situation in which a new internal event can occur. When no more internal events occur the process *settles down*, and we can infer a statement about the resulting stable state. Stable states are usually characterized by *state constraints*. In physics these states often minimize potential energy.

The next three sections discuss examples, a buzzer which has only internal events, the stuffy room scenario, and the blocks world.

3 Formalizing a buzzer

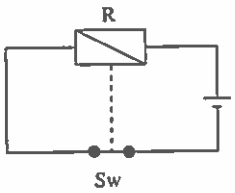


Figure 1: A buzzer.

Figure 1 displays a buzzer consists of a relay connected to a battery by a switch that is opened when the relay

operates. If the switch is on, the relay operates and opens the switch which turns off the relay which closes the switch. Thus the circuit oscillates and *never settles down* to a stable state.

The buzzer formalization has only internal events—at least once it is started, and this makes its operation easy to formalize.

State constraint axioms for formalizing a buzzer analogous to those often used for the stuffy room scenario would be immediately contradictory, asserting that the relay is on if and only if it is off. Our present situation calculus formalism follows human common sense reasoning directly and requires no special causal formalism or logic with implications not equivalent to their contrapositives.

There are effect axioms and occurrence axioms. The former are well known and give the effects of events. The latter assert that in situations in which certain fluents hold, certain events will occur.

We distinguish between the fluent On(Sw) asserting that the switch is on and the event Onn(Sw) that turns the switch on. The fluent holding in a situation is asserted by $Holds(\mathrm{On}(Sw), s)$. Likewise for the fluent On(R) and the event Onn(R) that concern the relay. We also have Off and Offf for the switch and the relay.

Effect axioms:

$$\begin{array}{l} Holds(On(R), Result(Onn(R), s)) \\ \neg Holds(On(R), Result(Offf(R), s)) \\ Holds(On(Sw), Result(Onn(Sw), s) \\ \neg Holds(On(Sw), Result(Offf(Sw), s)). \end{array} \tag{3}$$

Occurrence axioms:

$$\begin{array}{l} \neg Holds(On(Sw), s) \wedge Holds(On(R), s) \\ \qquad \rightarrow Occurs(Offf(R), s) \\ Holds(On(Sw), s) \wedge \neg Holds(On(R), s) \\ \qquad \rightarrow Occurs(Onn(R), s)) \\ Holds(On(R), s) \wedge Holds(On(Sw), s) \\ \qquad \rightarrow Occurs(Offf(Sw), s) \\ \neg Holds(On(R), s) \wedge \neg Holds(On(Sw), s) \\ \qquad \rightarrow Occurs(Onn(Sw), s) \end{array} \tag{4}$$

Note that each of the above occurrence axioms has a second term in the precondition. They are needed to avoid unwanted concurrent events.

Frame assertions—for now axioms:

$$\begin{array}{l} e = Onn(Sw) \vee e = Offf(Sw) \\ \qquad \rightarrow Holds(On(R), Result(e, s)) \\ \qquad\qquad \equiv Holds(On(R), s)). \end{array} \tag{5}$$

$$\begin{array}{l} e = Onn(R) \vee e = Offf(R) \\ \qquad \rightarrow Holds(On(Sw), Result(e, s)) \\ \qquad\qquad \equiv Holds(On(Sw), s)). \end{array} \tag{6}$$

These frame assertions tell what doesn't change. They are few enough in this case, since there are few actions and few fluents. In general it is more efficient to say what does change. In this case we have

$$\begin{array}{l} Changes(Onn(R), On(R), s), \\ Changes(Offf(R), On(R), s), \\ Changes(Onn(Sw), On(Sw), s), \quad \text{and} \\ Changes(Offf(Sw), On(Sw), s). \end{array} \tag{7}$$

In section 6 we describe how to get the frame assertions by circumscribing $Changes(e, f, s)$.

Let an initial situation, called $S0$, be given by

$$\neg Holds(On(Sw), S0) \wedge \neg Holds(On(R), S0) \tag{8}$$

We can proceed a step at a time. We have

$$Occurs(\mathrm{Onn}(Sw), S0) \tag{9}$$

in accordance with (4). Hence

$$Next(S0) = Result(Onn(Sw), S0), \tag{10}$$

and therefore, letting

$$S1 = Next(S0), \tag{11}$$

we have

$$\neg Holds(\mathrm{On}(R), S1) \wedge Holds(\mathrm{On}(Sw), S1). \tag{12}$$

Some elaborations of the buzzer axioms will be worth doing.

1. Allow the action of stopping the buzzer to occur at any situation.

2. Consider the action of stopping the buzzer as a concurrent event.

3. A concurrency elaboration along the lines of [McC92] and [MC98] might be to have two non synchronized buzzers B1 and B2 with no guaranteed temporal relation between the events involving B1 and B2.

4 The stuffy room scenario

A problem arises when the well-known stuffy room scenario is formalized with a state constraint that when both vents are blocked by pillows the room is stuffy and changes in fluents are minimized. This can lead

to the unintended model that when one vent is already blocked the action of blocking the other event causes the blocked vent to become unblocked in order to minimize change. Some complication of the formalism is required to deal with the phenomenon. Direct formalization in terms of actions and events avoids the difficulty. Also it corresponds better to the way we humans think about the problem, i.e. we think about the room becoming stuffy.

We use fluents Blocked1, Blocked2, and Stuffy. We have the action events Block1, Unblock1, Block2, Unblock2 and the internal events Getstuffy and Ungetstuffy. [1]

Effect axioms:

$$\begin{array}{l} Holds(Blocked1, Result(Block1, s)) \\ Holds(Blocked2, Result(Block2, s)) \\ \neg Holds(Blocked1, Result(Unblock1, s)) \\ \neg Holds(Blocked2, Result(Unblock2, s)) \\ Holds(Stuffy, Result(Getstuffy, s)) \\ \neg Holds(Stuffy, Result(Ungetstuffy, s)) \end{array} \tag{14}$$

Occurrence axioms:

$$\begin{array}{l} Holds(Blocked1, s) \wedge Holds(Blocked2, s) \\ \quad \wedge \neg Holds(Stuffy, s) \\ \quad\quad \rightarrow Occurs(Getstuffy, s) \quad \text{and} \\ (\neg Holds(Blocked1, s) \vee \neg Holds(Blocked2, s)) \\ \quad\quad \wedge Holds(Stuffy, s) \\ \quad \rightarrow Occurs(Ungetstuffy, s) \end{array} \tag{15}$$

The frame axioms are

$$\begin{array}{l} Changes(Block1, Blocked1, s), \\ Changes(Block2, Blocked2, s), \\ Changes(Unblock1, Blocked1, s), \\ Changes(Unblock2, Blocked2, s), \\ Changes(Getstuffy, Stuffy, s), \quad \text{and} \\ Changes(Ungetstuffy, Stuffy, s). \end{array} \tag{16}$$

[1] One of the referees suggested that using the fluents *Blocked*1 and *Blocked*2 and the corresponding actions was too special, and we should say that the room is stuffy when all the vents are blocked. We can accommodate his preference by introducing the vents as objects and using the axiom

$$\begin{array}{l} (\forall vent)(Holds(Blocked(vent), s)) \\ \quad \rightarrow Occurs(Getstuffy, s) \end{array} \tag{13}$$

and a corresponding axiom for the effect of unblocking a vent.

This is just a step towards a general commonsense theory of the effects of ventilation on stuffiness. Such a theory would have to take into account the fact that blocking the vents does not make the room stuffy under all circumstances. For now it's simpler to just consider the particular room with exactly two vents.

How they work is described in section 6.

We need to distinguish between *internal events* like $Getstuffy$ and *external events* like $Block1$. As we shall see, an external event may be an internal event of a more comprehensive narrative, e.g. one in which $Block1$ occurs when Mike is annoyed by cold air coming from $Vent1$.

We can tell a simple sequential story by first describing $S0$, e.g. by

$$\begin{array}{l} \neg Holds(Blocked1, S0) \wedge \neg Holds(Blocked2, S0) \\ \quad \wedge \neg Holds(Stuffy, S0). \end{array} \tag{17}$$

We can now write the narrative

$$\begin{array}{l} S1 = Result^*(Block1, S0) \\ S2 = Result^*(Block2, S1) \\ S3 = Result^*(Unblock2, S2) \\ S4 = Result^*(Block2, S3), \quad \text{etc.} \end{array} \tag{18}$$

Here $Result^*(e, s)$ is like the Rr of [McC95]. It is the result of doing a followed by the occurrence of whatever internal events occur. The assumption is that some sequence of internal events will occur after which the situation remains the same until another external event occurs. $Result^*(e, s)$ is undefined in the buzzer example in which internal events occur forever.

$Result^*$ requires an induction axiom or schema. Here's one candidate:

$$\begin{array}{l} P(s) \wedge (\forall s\ e)(P(s) \wedge Occurs(e, s) \rightarrow P(Result(e, s))) \\ \rightarrow P(Result^*(e, s)). \end{array} \tag{19}$$

The function $Next^*$ has the same relation to $Result^*$ that $Next$ has to $Result$. It gives the next situation to which no occurrence assertion applies. $Next^*$ satisfies

$$\begin{array}{l} Result^*(e, s) = Next^*(Result(e, s)), \\ (\forall e)(\neg Occurs(e, s)) \rightarrow Next^*(s) = s, \quad \text{and} \\ Occurs(e, s) \rightarrow Next^*(s) = Next^*(Result(e, s)). \end{array} \tag{20}$$

In the present case we will have

$$S1 = Result^*(Block1, S0) = Result(Block1, S0).$$

because no internal event will occur in $Result(Block1, S0)$. However, we'll have

$$\begin{array}{l} S2 = Result^*(Block2, S1) \\ \quad = Result(Getstuffy, Result(Block2, S1)), \end{array} \tag{21}$$

because now the internal event $Getstuffy$ will occur. Thus we'll have $\neg Holds(Stuffy, S1)$ but $Holds(Stuffy, S2)$, $\neg Holds(Stuffy, S3)$, and $Holds(Stuffy, S4)$.

We can write

$$\begin{aligned} S4 &= Result^*(Block2, Result^*(Unblock2, \\ &\quad Result^*(Block2, Result^*(Block1, S0)))) \\ &= Result(Getstuffy, Result(Block2, \\ &\quad Result(Ungetstuffy, Result(Unblock2, \\ &\quad Result(Getstuffy, Result(Block2, \\ &\quad\quad Result(Block1, S0))))))), \end{aligned} \tag{22}$$

which can also be written

$$\begin{aligned} S4 &= Result^*(Block1; Block2; Unblock2; Block2, S0) \\ &= Result(Block1; Block2; Getstuffy; Unblock2; \\ &\quad Ungetstuffy; Block2; Getstuffy, S0) \end{aligned} \tag{23}$$

Here we extend the meaning of *Result* to allow a sequence of events as an argument.

4.1 Telling stories using *Occurs* and *Next*

Another way of telling stories is to always use *Occurs*. An external event is axiomatized by asserting that it occurs.

The above story is then given by

$$\begin{aligned} &Occurs(Block1, S0) \\ &S1 = Next(S0) = Result(Block1, S0) \\ &Occurs(Block2, S1) \\ &S1' = Next(S1) = Result(Block2, S1) \\ &Occurs(\text{Getstuffy}, S1'), \text{by inference} \\ &S2 = Next(S1') = Result(Getstuffy, S1') \\ &Occurs(Unblock2, S2) \\ &S2' = Next(S2) = Result(Unblock2, S2) \\ &Occurs(Ungetstuffy, S2') \text{by inference} \\ &S3 = Next(S2') = Result(Ungetstuffy, S2') \\ &Occurs(Block2, S3) \\ &S3' = Next(S3) = Result(Block2, S3) \\ &Occurs(Getstuffy, S3') \text{by inference} \\ &S4 = Next(S3') = Result(Getstuffy, S3'). \end{aligned} \tag{24}$$

We can also write the story more briefly as

$$\begin{aligned} &Occurs(Block1, S0) \\ &S1 = Next^*(S0) = Result^*(Block1, S0) \\ &Occurs(Block2, S1) \\ &S2 = Next^*(S1) = Result^*(Block2, S1) \\ &Occurs(Unblock2, S2) \\ &S3 = Next^*(S2) = Result^*(Unblock2, S2) \\ &Occurs(Block2, S3) \\ &S4 = Next^*(S3) = Result^*(Block2, S3) \end{aligned} \tag{25}$$

Still more briefly

$$\begin{aligned} S4 &= Next^*(Next^*(Next^*(Next^*(S0)))) \\ &= Result^*(Block2, Result^*(Unblock2, \\ &\quad Result^*(Block2, Result^*(Block1, S0)))) \end{aligned} \tag{26}$$

4.2 Two elaborations of the stuffy room scenario

The first elaboration says that when Pat finds the room stuffy he unblocks vent2. We have

$$Holds(Stuffy, s) \rightarrow Occurs(Does(Pat, Unblock2), s), \tag{27}$$

or, more elaborately,

$$\begin{aligned} &Holds(Stuffy, s) \wedge \neg Holds(Uncomforable\text{-}Pat, s) \\ &\quad \rightarrow Occurs(Becomes\text{-}Uncomfortable(Pat), s), \\ &Holds(Uncomfortable, Pat, \\ &\quad Result(Becomes\text{-}Uncomforable(Pat), s)) \\ &Holds(Uncomfortable, Pat, s) \\ &\rightarrow Occurs(Does(Pat, Unblock\text{-}Vent2), s), \\ &\neg Holds(Blocked2, Result(Does(Pat, Unblock\text{-}Vent2), s)). \end{aligned} \tag{28}$$

(24) remains the same except that perhaps we should change the notation so that instead of $S3$ and $S3'$ we write $S2''$ and $S2'''$, since these are now intermediate situations. The situation $S4$ is now unstable.

Now let's add a second elaboration in which Mike finds the room cold when there is an unblocked vent and blocks vent2. It is expressed by adding

$$Holds(Unstuffy, s) \rightarrow Occurs(Does(Mike, Block2), s). \tag{29}$$

With both of these elaborations, we get an oscillation; Pat unblocks vent2 and Mike blocks it again. *Result** and *Next** are no longer defined.

5 The blocks world

Assume enough unique names axioms.

The blocks world involves the frame problem in a more significant way than do the buzzer and the stuffy room scenarios.

We use the predicate $Prevents(p, e, s)$ to say that a move is prevented by there being a block on top of the block to be moved or on the destination unless the destination is the table. We thereby skip the use of the fluent $Clear(x)$ prevalent in many blocks world sitcalc theories.

Here's the effect axiom for moving a block.

$$(\forall p)(\neg(Prevents(p, Move(x,y), s) \wedge Holds(p,s)))$$
$$\rightarrow$$
$$(Holds(On(x,y), Result(Move(x,y),s)))$$
$$\wedge$$
$$((Holds(On(x,z),s)) \wedge z \neq y$$
$$\rightarrow \neg Holds(On(x,z), Result(Move(x,y),s)))), \quad (30)$$

and here are the axioms for prevention:

$$Prevents(On(z,x), Move(x,y), s) \quad \text{and}$$
$$y \neq Table \rightarrow Prevents(On(z,y), Move(x,y), s). \quad (31)$$

We adopt the usual way of emphasizing the frame problem by introducing the action of painting a block a certain color. Thus

$$(\forall p)(\neg(Prevents(p, Paint(x,c), s) \wedge Holds(p,s)))$$
$$\rightarrow Holds(Color(x,color), Result(Paint(x,color),s)), \quad (32)$$

or, using object valued, i.e. non propositional, fluents,

$$(\forall p)(\neg(Prevents(p, Paint(x,c), s) \wedge Holds(p,s)))$$
$$\rightarrow Value(Color(x),$$
$$Result(Paint(x,color),s)) = color. \quad (33)$$

The change axioms for the blocks world are

$$Changes(Paint(x,c), Color(x), s),$$
$$Holds(On(x,z),s)$$
$$\rightarrow Changes(Move(x,y), On(x,z), s) \quad (34)$$
$$\wedge Changes(Move(x,y), On(x,y), s).$$

The nonmonotonic reasoning associated with the blocks world will be discussed after the section dealing with nonmonotonic reasoning in situation calculus in general.

6 Nonmonotonic reasoning—situation by situation

We use circumscription to minimize the events that occur in a situation, the fluents that might prevent an event from having its standard effect, and the changes in fluents. In contrast to the formalism of [McC86] which minimized predicates over all the arguments, we minimize for each successive situation separately. However, in doing this minimization in s we take as fixed the $Holds(f,s)$ sentences and the $Value(exp,s) = \ldots$ sentences inferred from the effects of the event that led to the situation. We are giving up the possibility of trading and abnormality in one situation for an abnormality in another.

Doing the nonmonotonic reasoning in situations successively corresponds to the way people predict the consequences of sequences of actions and events. It seems to give the same conclusions as Yoav Shoham's chronological minimization [Sho88] but is computationally more straightforward. Like chronological minimization, it avoids the Yale shooting problem and its friends.[2]

However, we advocate this only for projection problems, i.e. reasoning about the future from information about the past. The method is not appropriate for the *stolen car scenario* in which one has to reason from an assertion (that the car is missing) about a later situation. [3]

With the present formalism, the person or agent setting up the problem must know that projection forward in time is appropriate. It would be better if this were a consequence of the formalized facts.

Now let's consider circumscribing at each situation separately. The simplest case is when we have a predicate $Foo(x,y,s)$.

We write the axioms

$$Foo' \leq_s Foo \equiv (\forall x\ y)(Foo'(x,y,s) \rightarrow Foo(x,y,s)),$$
$$(Foo' <_s Foo) \equiv (Foo' \leq_s Foo) \wedge \neg(Foo' =_s Foo),$$
$$Foo' =_s Foo \equiv (\forall x\ y)(Foo'(x,y,s) \equiv Foo(x,y,s)). \quad (35)$$

Then the circumscription of $Foo(x,y,s)$ takes the form

$$Axiom(Foo, vars, s) \wedge (\forall foo'\ vars')(Axiom(foo', vars')$$
$$\rightarrow \neg(foo' <_s Foo)). \quad (36)$$

Here *vars* stands for a list of the entities being varied as Foo is minimized.

[2] The ideas of internal and external events of the preceding sections are independent of the formalism used for nonmonotonic reasoning. For example, Golog [Rei01] or the Causal Calculator [aA01] could be used—perhaps with some modifications for the buzzer and the oscillating stuffy room.

[3] Actually part of the stolen car scenario can be treated provided we don't suppose that the car being missing is to be projected from information about the past. Certainly we can go forward from the situation in which the car is missing to further events in the future. Likewise, in the story of Junior's travels [McC92], we can assert that Junior loses his ticket to Moscow in London and reason forward from that fact.

This spells out to

$$\begin{array}{l} Axiom(Foo, vars, s) \wedge (\forall foo'\ vars') \\ (Axiom(foo', vars') \wedge ((\forall x\ y)(foo'(x, y, s) \\ \qquad \rightarrow Foo(x, y, s)) \\ \rightarrow (\forall xy)(Foo(x, y, s) \equiv foo'(x, y, s)))). \end{array} \tag{37}$$

Call this formula $Circ(Axiom; Foo; vars; s)$. This is the notation of [Lif94] with the addition of the argument s to say that s is kept fixed.

The general frame axioms are

$$\neg Changes(e, p, s) \rightarrow (Holds(p, Result(e, s)) \equiv Holds(p, s)) \tag{38}$$

for propositional fluents and

$$\neg Changes(e, f, s) \rightarrow Value(f, Result(e, s)) = Value(f, s). \tag{39}$$

for general fluents.

Suppose we allow complex fluents, say $p\ And\ q$ when p and q are propositional fluents. We then need an axiom

$$Changes(e, p, s) \vee Changes(e, q, s) \rightarrow Changes(e, p\ And\ q, s). \tag{40}$$

Similar axioms are required for the other propositional functions of fluents and for the compositions of non-propositional fluents.

[This leads to difficulties when we want to delimit what changes, since there are arbitrarily complex compositions of fluents. We'll confine ourselves to elementary fluents for now by not putting compositions in the language.]

In these circumscriptions we also minimize $Holds$.

This tolerates elaborations like

$$Holds(Weak, s) \rightarrow Prevents(Weak, Move(x, y), s). \tag{41}$$

If $Holds(Weak, s)$ isn't asserted, $Move(x, y)$ will not be prevented.

Lin and Shoham, [LS95] consider a theory of action to be *provably correct* if doing the nonmonotonic reasoning results in a complete nonmonotonic theory of the action. This seems like a worthy goal, but I don't know if the present theory achieves it.

7 Actions and other events

The previous sections presented a formalism adequate for the examples discussed. In this section we discuss the situation calculus in general and its connection with the real world. We also discuss relations between different situation calculus theories, e.g. theories at different levels of detail, [4] with actions by agents as a special case. Thus an action term a is considered an abbreviation of the event term $Does(person, a)$. Besides *effect axioms* formalizing $Result(e, s)$ [$do(e, s)$ in Canada and its colonies], there are *occurrence axioms* asserting that in situations satisfying certain expressions in the fluents, an event e occurs—written $Occurs(e, s)$. [5]

Before giving effect and occurrence axioms, we present some general considerations concerning situation calculus and its applications.

7.1 Situation calculus and the real world

There have been many formulations of situation calculus.

[MH69] regarded a situation as a snapshot of the world at some instant of time. Such a system could not be known and described completely, but a person or program could know facts about a situation, i.e. the values of some fluents, and could infer some consequences of some actions from these facts. Situations are examples of *rich entities*, i.e. entities involving more detail than can be specified. *Poor entities* have finitely describable structures.

However, theories of action and change[6] often use a more limited notion of situation. Thus Raymond Reiter [Rei01] and his colleagues regard situations as the nodes of a tree based at an initial situation $S0$ and whose edges branching from a situation s are the ac-

[4] [McC59] proposed mathematical logic as a tool for representing facts about the consequences of actions and using logical reasoning to plan sequences of actions that would achieve goals. Situation calculus as a formalism was proposed in [McC63] and elaborated in [MH69]. The name "situation calculus" was first used in [MH69] but wasn't defined there. [McC86] proposed to solve the frame and qualification problems by circumscription, but the proposed solution to the frame problem was incorrect. [Sha97] and [Rei01] describe several situation calculus formalisms and give references.

[5] I suspect I need to pound the table a little here. Actions are just a kind of event, and formalized *reasoning about actions and change* need to treat events as the general case and those events which are actions as special. This has long seemed obvious to me, but I find that many other researchers don't want to use the same formalism for events that are not actions of agents and those which are.

The consequence has been the introduction of extensions to logic for treating what are called domain constraints, most of which are better treated by formalizing events.

[6] "events and change" would be better terminology

tions that may be taken in s. Other researchers, including Murray Shanahan [Sha97] and myself, use $S0$ as just a name for some situation whose consequences are of interest.

The viewpoint of this article is that a situation s is arbitrary element of a space $Sits$ of situations, i.e. s bears the same relation to $Sits$ as a group element bears to a group. Situation calculus theories relate situations, fluents and action by axioms, i.e. are abstract structures satisfying the theory.

A robot can use a *poor* situation calculus theory T to decide what to do in a world of rich situations. For example, the robot's blocks world theory may only allow specifying that one block is on another, not where it is located on the other. Suppose we have a mapping $Observe$ from a subset of rich situations to poor situations. When the robot observes a world situation s to which the theory T applies, it obtains a poor situation $Observe(s) \in Sits(T)$. Using the theory T, the robot infers that a certain action a will advance its goal. It then performs an action $Execute(a)$ in the world. If the theory T corresponds to the world properly, $Result(Execute(a), s)$ will be an improved situation.

It isn't the purpose of this paper to develop a theory of the correspondence between rich real world situations and those of limited sitcalc domains. However, the way we formalize sitcalc is motivated by the hope of making these correspondences in a later theory.

Whether an event is external depends on the theory. If we can formulate when an event e will occur, then we can make our theory more powerful by including an occurrence axiom for that event. If we assume a deterministic world, the limiting case is a theory in which all events are internal.

8 Elaboration tolerance

An important feature of human common sense is that human knowledge of a phenomenon is often readily elaborated to take new information into account. It is important that logical theories of common sense phenomena also have this property. [McC99] has a detailed discussion.

Situation calculus theories benefit from several kinds of elaboration. Section 4 discusses elaborating the stuffy room theory by adding occurrence axioms for a person being motivated to open a vent when the room becomes stuffy. [McC92] constructs a theory of a persons travel planning which can be elaborated by adding a sentence asserting that he loses his airplane ticket at a certain point in his journey. Because the reasoning depends on minimizing occurrences, we can no longer conclude that the original travel plan will succeed.

In general, elaboration tolerance concerns making it easy to modify a theory, but the simplest kind of elaboration is to add one or more sentences to an existing theory. It is desirable that elaborations be doable in this way as much as possible. [McC99] discusses when this can and cannot be done for a given theory and how to make theories for which elaboration by conjoining sentences is possible. Theories expressed in natural language have this kind of elaboration tolerance to a high extent.

9 Extensions of the formalism and problems they present

The basic situation calculus admits many useful extensions. The ideas of this section are tentative.

9.1 Concurrency

There are two limiting cases of concurrency that can be treated in the situation calculus.

Easy concurrency:

Two or more events, say e_1 and e_2 occur in a situation s and result in the same next situation $Next(s)$. The fluents that hold in $Next(s)$ are those determined by the effect axioms for $e1$ and $e2$ separately. Thus if we move a block and paint it concurrently, it will have both the new location and the new color in $Next(s)$.

General concurrency:

Two *processes*, starting, say from initial situations $S0$ and $S0'$ take place and affect different sets of fluents. If nothing is said about the timing of the processes and no axioms of interaction are given, nothing can be inferred about the relative timing of the processes. Moreover, what can be inferred about the values of the fluents in successive situations is exactly what can be inferred by the processes taken separately. Thus Louis Pasteur was elected to the French Academy of Sciences in 1862 concurrently with certain battles of the American Civil War, but historians mention neither process in connection with the other. This is a limiting case, i.e. the case of zero interaction. Two theories of separate processes can be combined by taking the conjunction of their axioms. The combined theory is a *conservative extension* of each separate theory. It can be useful to elaborate the combined theory by giving axioms for the interaction. [McC95] and [MC98] treat elaborating theories of two non-interacting processes

by adding axioms of interaction. Those articles treat Junior traveling in Europe and Daddy stacking gold blocks in New York. There is no interaction until we adjoin assertions about Junior losing an airplane ticket and asking Daddy for money, thus forcing Daddy to sell one of the blocks he was stacking.

We hope to combine the ideas of the two above-mentioned articles with those of this article in future work.

9.2 Events whose occurrence depends on the past

Suppose we want George to unblock both vents when the room becomes stuffy. When he has unblocked one vent, the room becomes unstuffy, so the physical situation is as it was when he blocked the first vent, so he needs to remember that the room was previously stuffy. We can make occurrences depend on past situations by adding for each event e an additional effect axiom

$$Past(Result(e, s)) = s. \tag{42}$$

Notice that $Past(Past, s))$ is the situation two events back.

We can have George unblock Vent1 after he has unblocked Vent2 and the room has become unstuffy by introducing the occurrence axiom

$$Stuffy(Past(Past(s)) \rightarrow Occurs(Unblock1, s). \tag{43}$$

The history as just described does not say what events occurred. This information is provided by having for each event e the axiom

$$Lastevent(Result(e, s)) = e. \tag{44}$$

Notice that this formalization is noncommittal as to whether the information is in an actor's memory.

This seems neat, and maybe it will be useful.

9.3 "Branching time" and "linear time"

We can tell a story by saying what occurs in each situation. In some situations what occurs is determined by an occurrence action and depends on the fluents holding in the situation. In other situations, we simply provide an axiom, e.g. $Occurs(Birth(\text{Benjamin-Franklin}), S1806)$. This is a linear time theory.

However, linear time and branching time are sometimes appropriately used together. Suppose we wish to say that the actor will take the low road or the high road according to which will get him to Scotland first. We can write

$$\begin{aligned}&\textbf{if } [\text{Arrival-time}(Result(\text{Take-Low-Road}, s))\\ &\qquad \leq \text{Arrival-time}(Result(\text{Take-High-Road}, s))]\\ &\textbf{then } Occurs(\text{Take-Low-Road}, s)\\ &\textbf{else } Occurs(\text{Take-High-Road}, s).\end{aligned} \tag{45}$$

Here we have used a branching time criterion for a linear time action.

This is not as elaborate as actual human behavior in which mental events occur calculating which route will lead to earliest arrival.

9.4 Induction in the situation calculus

Several kinds of mathematical induction seem to be required. For example, one may want to prove a proposition $P(Next^*(s))$ by showing that it is true for s and is preserved by the events that occur between s and $Next^*(S)$. A related kind of induction is needed to prove that something is true for all situations arising in the operation of a buzzer. The simplest case of the $Next^*$ induction might be to show that a block unmoved by each of a sequence of events is in the same position in $Next^*(s)$.

The simplest situation calculus is Reiter's [Rei01]. The formula is

$$[P(S0) \wedge ((\forall a\ s)(P(s) \rightarrow P(Result(a, s))))] \rightarrow (\forall s)P(s). \tag{46}$$

Here are two formulas

$$\begin{aligned}&[P(s) \wedge ((\forall e\ s)(P(s) \wedge Occurs(e, s) \rightarrow P(Next(s))))]\\ &\rightarrow P(Next^*(s)).\end{aligned} \tag{47}$$

(47) is appropriate when $Next^*(s)$ is defined.

When $Next^*(s)$ is not defined, as in the buzzer case, we can use $s \leq s'$ to mean that s' is a distant successor of s and have the axiom.

$$\begin{aligned}&[P(s) \wedge s \leq s'\\ &\qquad \wedge((\forall e\ s)(P(s) \wedge Occurs(e, s) \rightarrow P(Next(s))))]\\ &\rightarrow P(s').\end{aligned} \tag{48}$$

9.5 Formalizing Oscillations

The buzzer oscillates, i.e. the situation repeats again and again. So does the stuffy room scenario with the two elaborations that cause Vent2 to become blocked

and unblocked repeatedly. However, we don't need a complete repetition of the situation to have oscillation. Suppose, or example, we add a clock to the buzzer, a natural number valued fluent that each event increments by 1. Then although the whole situation would not repeat, we would still want to consider the system as oscillatory.

This suggests a relative notion of oscillatory, i.e. oscillatory with respect to certain fluents.

Moreover, we would like to consider the buzzer as oscillating even if we provide for it stopping its oscillation by being turned off.

As we have described the buzzer, it cannot be turned off. Likewise the stuffy room process cannot be changed once we have added the elaborations about people blocking and unblocking the vent. See (27) and (29).

Here's a way of putting interventions into the formalism.

Let a be an action, e.g. stopping that damn buzzer. The following two axioms describe an elaboration that interpolates an action after a normal internal action. In the buzzer case it would be opening an additional switch in the circuit. The additional switch isn't in Fig. 1 or described in section 3.

$$Occurs(a,s) \wedge External(a) \wedge Occurs(e,s) \\ \rightarrow Next(s) = Result(a, Result(e,s)) \tag{49}$$

and

$$Occurs(e,s) \wedge (\forall e')(Occurs(e',s) \rightarrow e' = e) \\ \rightarrow Next(s) = Result(e,s). \tag{50}$$

This is a limited kind of concurrency. Only certain kinds of interventions can be done this way.

9.6 State constraints after all

As was shown in Section 4, the condition for a room being stuffy is better formalized with effect axioms, occurrence axioms, and the events $Getstuffy$ and $Ungetstuffy$. Lin and Reiter [LR94] consider the Emperor's decree that no more than one object (block) be yellow, which may be regarded as a domain constraint. They point out that it is more efficient to encode the constraint as a precondition that a block may be painted yellow only if no block is already yellow. Their way of expressing this does not readily elaborate to require that no more than seven blocks be yellow.

I think logical AI needs a more complex treatment. It seems to me that efficiency conflicts with generality. It is bad or dangerous to have more than one yellow block, but perhaps only if one is not a special favorite of the emperor or if one is just about to die anyway. The point is that common sense (at least human level common sense) requires that such constraints tolerate elaboration. Human level common sense also allows the constraint to become an action precondition as a result of some inference. This inference should take place within the logical formalization.

Lin and Reiter include the following formula.

$$(\forall x\ y\ s)(Poss(Paint(x,y),s) \\ \equiv (Nearby(x,s) \wedge Haspaint(y,s) \\ \wedge(\forall x_1)(Color(x_1, Yellow, s) \wedge y = Yellow \rightarrow x = x_1))). \tag{51}$$

This formula is specialized to the emperor tolerating just one yellow block. If he tolerates 7 yellow blocks, we had better use set notation, i.e. refer to $card(\{x|Color(x,Yellow)\}) \leq 7$.[7]

There are some domain constraints that are not naturally formalized by internal actions. One is the blocks world constraint that a block may not be on top of itself. Formulas like

$$Above(Top(block), Bottom(block), s) \tag{52}$$

or even

$$Height(Top(block),s) - Height(Bottom(block),s) \\ \geq 1.0cm \tag{53}$$

tell more about the world than the simple

$$\neg On(block, Top(block), s). \tag{54}$$

An important application for the direct use of state constraints is when an event starts a process that eventually leads to an equilibrium state. For example, if I drop a coin on the floor it will bounce around for a while and then settle down. It will reach equilibrium in a second or so, and I am interested in whether the coin ends up heads or tails rather than in the process of its settling down. In the case of the coin the equilibrium condition, at least what we want to know about it, is easy to state, namely

$$On(coin, floor, Result^*(Drop(coin,s))) \\ \wedge(Heads(coin, Result^*(s)) \\ \vee Tails(coin, Result^*(s))), \tag{55}$$

where using $Result^*$ means that we are skipping by some internal events, in this case not formalized.

[7] I pound the table here because of some resistance to the idea that axiomatic set theory makes logical AI easier.

Another example may be concocted from the elaborated stuffy room scenario. While Pat and Mike disagree in their preferences, under normal circumstances we can suppose they will come to an agreement in some short time. One will defer to the other in the matter of the blocked vents. As with the coins, the theory of eventual agreement doesn't predict what the agreement will be.

More generally, Aarati Parmar suggests that internal events are evoked by any non-equilibrium situations.

9.7 Javier Pinto's formalism

The work closest to the present is [Pin98b], as one of the referees forcefully pointed out. There are substantial differences, both in approach and in the formalisms motivated by the different approaches.

Pinto uses the Reiter notion of situations as trees built from the initial situation $S0$ by iterations of forming $do(a, s)$ where a is an action and s a previously formed situation term.

Pinto (as does Reiter) builds time, represented by a real number, into his situation calculus formalism. It seems to me that making time fit the tree structure of situation terms leads to complications. Pinto has five different *occur* predicates, whereas we have only one. His occurrence axioms all have time parameters. Our occurrence axioms involve only situations and fluents and are therefore simpler. The examples of the present article do not involve time explicitly. When time must be explicit, we propose to treat the passage of time as an independent situation calculus process running concurrently with the processes we are treating.

Pinto includes the following interesting examples. We show how our method treats a few of them.

1. "The sun will rise tomorrow at 6:03 am." Here we have two concurrent processes: the passage of time and the path of the sun through the sky. The sentence describes an interaction.

We can represent the sentence by

$$Value(Time, s) = Time(Tomorrow603am) \rightarrow Occurs(Sunrise, s),$$

where we are not taking into account the explicit indexical of tomorrow, and the implicit indexical that sunrise being a 6:03am must refer to a specific latitude and longitude.

2. "If you eat the forbidden fruit you will be expelled." Pinto treats this as one event causing another but remarks that it might be better formalized as a state, i.e. that of having eaten the fruit, giving rise to an event. That's how the present paper would treat it, i.e.

$$Holds(Has\text{-}occurred(Eat(Forbidden\text{-}fruit)), s) \rightarrow Occurs(Does(God, Expel(Eater)), s), \tag{56}$$

together with the general moving finger axioms

$$Occurs(e, s) \rightarrow Holds(Has\text{-}occurred(e, Next(s))),$$
$$\text{and } Holds(Has\text{-}occurred(e, s) \wedge s < s' \rightarrow Holds(Has\text{-}occurred(e, s')). \tag{57}$$

3. "The train to Ottawa leaves every day at 7 pm." where it is understood that this scheduled event may not occur under exceptional circumstances.

$$Value(Time, s) = Time(7pm) \wedge \neg Prevented(Train\text{-}Leaves\text{-}for\text{-}Ottowa, s) \rightarrow Occurs(Train\text{-}Leaves\text{-}for\text{-}Ottowa, s).$$

4. "If my neighbor's burglar alarm goes off while I am at home, I will call the police." Pinto treats this example and the previous one by slightly different formalisms, one involving a predicate $occurs_{po}(action, time)$ and the other a predicate $occurs_{ct}(action, time, action2)$.

5. The Miller-Shanahan [RM94] example of the briefcase.

6. "My house has a burglar alarm. If the alarm is connected, I have exactly 60 seconds to deactivate it after opening the main door. If I am unable to disconnect the alarm, it will go off."

7. "Upon an insertion into EMP or an update to EMP, the new SAL is checked, and if it exceeds \$100,000, then the JobTitle of this employee is added to HPAID, assuming it was not there already."

$$Holds(Checksalary, Result(Insert(EMP, y, s))) \wedge Holds(Checksalary, Result(Update(EMP, y, s))).$$

$$Holds(Checksalary(employee, s) \rightarrow Occurs(Add(JobTitle(employee), HPAID), s)$$

[Pin98a] introduces $occurs(a, s)$, where a is a "natural action". Natural actions partly correspond to internal events. The article is dedicated to concurrent events, to which I hope devote a separate article.

10 Concluding remarks

Events that are not actions have been previously used—at least by Fangzhen Lin [Lin98], Sheila McIlraith [McI00], and Javier Pinto.

Occurrence axioms are even more important in the treatment of concurrent events in situation calculus—to be the subject of another article.

This work benefited from discussions with Eyal Amir, Tom Costello, Ron Fadel, Hector Levesque, Vladimir Lifschitz, Fangzhen Lin, Sheila McIlraith, Leora Morgenstern, Aarati Parmar, Raymond Reiter, and Tran Son and the comments of three anonymous referees.

This research was partly supported by SRI Subcontract No. 34-000144 under SPAWAR Prime Contract No. N66001-00-C-8018.

References

[aA01] Texas Action Group at Austin. Causal calculator home page, 2001. http://www.cs.utexas.edu/users/tag/cc.

[GS88] Matthew L. Ginsberg and David E. Smith. Reasoning about action I: A possible worlds approach. *Artificial Intelligence*, 35(2):165–195, 1988.

[Lif94] Vladimir Lifschitz. Circumscription. In J. A. Robinson Dov M. Gabbay, C. J. Hogger, editor, *Handbook of logic in artificial intelligence and logic programmin*, volume 3, pages 297–352. Oxford, 1994.

[Lin98] Fangzhen Lin. On the relationships between static and dynamic causal rules in the situation calculus. In Charles L. Ortiz, Jr., editor, *Working Notes of the AAAI Spring Symposium on Prospects for a Commonsense Theory of Causation*, pages 38–43, Menlo Park, CA, 1998. American Association for Artificial Intelligence.

[LR94] Fangzhen Lin and Ray Reiter. State constraints revisited. *Journal of Logic and Computation*, 4:655–678, 1994.

[LS95] Fangzhen Lin and Yoav Shoham. Provably correct theories of action. *Journal of the ACM*, 42(2):293–320, March 1995.

[MC98] John McCarthy and Tom Costello. Combining narratives. In *Proceedings of Sixth Intl. Conference on Principles of Knowledge Representation and Reasoning*, pages 48–59. Morgan-Kaufman, 1998.

[McC59] John McCarthy. Programs with Common Sense[8]. In *Mechanisation of Thought Processes, Proceedings of the Symposium of the National Physics Laboratory*, pages 77–84, London, U.K., 1959. Her Majesty's Stationery Office. Reprinted in [McC90].

[McC63] John McCarthy. Situations, actions and causal laws. Technical Report Memo 2, Stanford University Artificial Intelligence Laboratory, Stanford, CA, 1963. Reprinted in [Min68].

[McC86] John McCarthy. Applications of Circumscription to Formalizing Common Sense Knowledge[9]. *Artificial Intelligence*, 28:89–116, 1986. Reprinted in [McC90].

[McC90] John McCarthy. *Formalizing Common Sense: Papers by John McCarthy.* Ablex Publishing Corporation, 1990.

[McC92] John McCarthy. Overcoming unexpected obstacles[10]. Web only, 1992.

[McC95] John McCarthy. Situation Calculus with Concurrent Events and Narrative[11]. 1995. Web only, partly superseded by [MC98].

[McC99] John McCarthy. Elaboration tolerance[12]. *web only for now*, 1999.

[McI00] Sheila A. McIlraith. An axiomatic solution to the ramification problem (sometimes). *Artificial Intelligence*, 116(1–2):87–121, 2000.

[MH69] John McCarthy and Patrick J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence[13]. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969. Reprinted in [McC90].

[Min68] Marvin Minsky, editor. *Semantic information processing.* MIT Press, 1968.

[Pin98a] Javier A. Pinto. Concurrent actions and interacting effects. In Anthony G. Cohn, Lenhart Schubert, and Stuart C. Shapiro, editors, *KR'98: Principles of Knowledge Representation and Reasoning*, pages 292–303. Morgan Kaufmann, San Francisco, California, 1998.

[Pin98b] Javier A. Pinto. Occurrences and narratives as constraints in the branching structure of the situation calculus. *Journal of Logic and Computation*, 8(6):777–808, 1998.

[Rei01] Raymond Reiter. *Knowledge in Action.* M.I.T. Press, 2001.

[RM94] R.S.Miller and M.P.Shanahan. Narratives in the situation calculus. *Journal of Logic and Computation*, 4(5):513–530, 1994.

[Sha97] Murray Shanahan. *Solving the Frame Problem, a mathematical investigation of the common sense law of inertia.* M.I.T. Press, 1997.

[Sho88] Yoav Shoham. Chronological ignorance: Experiments in nonmonotonic temporal reasoning. *Artificial Intelligence*, 36(3):279–331, 1988.

[8] http://www-formal.stanford.edu/jmc/mcc59.html

[9] http://www-formal.stanford.edu/jmc/applications.html

[10] http://www-formal.stanford.edu/jmc/glasgow.html

[11] http://www-formal.stanford.edu/jmc/narrative.html

[12] http://www-formal.stanford.edu/jmc/elaboration.html

[13] http://www-formal.stanford.edu/jmc/mcchay69.html

Invited Speakers and Panel Abstract

The Role of Higher Order Similarity in Induction and Concept Formation

Peter Gärdenfors
Lund University Cognitive Science,
Kungshuset, Lundagård,
S-222 22 Lund, Sweden
Peter.Gardenfors@lucs.lu.se

Abstract

Traditional symbol based theories of induction and concept formation have built on the following two assumptions (among others):

1. There is a set of primitive properties (predicates), from which new concepts are constructed and correlations between properties are determined.
2. Correlations between predicates are described by conditional probabilities.

However, the first assumption does not answer the question of how we can learn the primitive predicates in the first place. To handle this problem, machine learning algorithms or models based on artificial neuron networks have been used instead.

The second assumption has been motivated within the Bayesian tradition as being the most rational way of determining correlations between properties. The use of conditional probabilities is motivated by the principle of maximal entropy (Williams 1980). However, by refering to maximal entropy all instances of a concept are treated as unrelated. On the other hand, in human concept formation and inductive reasoning, judgments of similarity are essential. Similarity cannot be handled in a natural way in terms of Bayesian conditional probablities.

I will present a model, developed together with Christian Balkenius, of how (higher order) similarities can be used in induction and concept formation. The model is based on a layered structure of self-organising maps (Kohonen 1995). The model will be related to the conceptual spaces developed in my recent book (Gärdenfors 2000). It will be shown how the model can handle some facts concerning human inductive reasoning that have been problematic for models based on conditional probabilities. I will also connect the model to the theory of case based reasoning proposed by Gilboa and Schmeidler.

The Semantic Web: KR's Worst Nightmare?

Jim Hendler
Dept. of Computer Science
A.V. Williams Bldg.
University of Maryland
College Park, MD 20742 USA
hendler@cs.umd.edu

Abstract

For many years the knowledge representation community has worked with a set of assumptions that have led to a solid body of work - formal, clean, and largely irrelevant to the world at large. Expressivity, consistency and decidability have been the guiding principals of the field, with little worry about actual performance, scaling or usability issues. In short, the field has flourished on theoretical elegance and incremental results, with little perturbation over the past decade or so.

Recently however, a new wind has been blowing through the knowledge representation world. The World Wide Web has been seeing a growing demand for semantics! Large collections of web pages, images and collections need organizing principals for management. Databases and web services must be able to advertise capabilities and find each other. XML datasets and schemas need to be linked in scalable ways that allow a WEB of semantic information to emerge.

Unfortunately, most of those demanding knowledge representation on the web have different goals than those of the KR community. Scalability and performance trump decidability, ease of use fights with expressivity in language design, and consistency is impossible to guarantee, if it is even desirable. In short, the world is demanding KR, but it refuses to agree to the methodology of our community!

Other researchers, such as those in hypertext and information retrieval have seen their fields changed beyond recognition as the web reached into their communities. KR is next! -- but are we up to the challenge?? In this talk, I will describe the needs of this new world of web semantics, and challenge the KR community to rise to the historic opportunity to help address them.

The Philosophical Soccer Player

Bernhard Nebel
Institut für Informatik
Albert-Ludwigs-Universität Freiburg
Georges-Köhler-Allee, Geb. 52
D-79110 Freiburg, Germany
nebel@informatik.uni-freiburg.de

Abstract

The main task of a soccer player is to score goals. However, there are moments in life when questions like the following become relevant: Is the ball I am seeing a hallucination or is it real? Should I revise my beliefs about where the ball is? And if so, what is the next action I should execute? Would this action be to the benefit of my team? In the talk I will address these questions and show how one can create a successful robotic soccer team by giving the right answers.

Panel
Are Upper-Level Ontologies worth the effort?

Chair

Christopher Welty

Panelists

Pat Hayes **Jim Hendler** **Nicola Guarino** **Fritz Lehmann**

Abstract

Upper level ontologies are envisioned by some as being a necessary first step for delivering the kind of interoperability that ontologies promise, and can also serve as a starting point when building a new domain-specific ontology. This yields a requirement for upper-level ontologies that they be as universal and as correct as possible; two goals that have thus far made the effort to develop them extremely expensive. Some claim the expense does not justify the perceived benefit, still others argue that upper-level ontologies are a pipe dream that can never be achieved. The purpose of this panel will be to debate this issue.

Author Index

PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING: PROCEEDINGS OF THE EIGHTH INTERNATIONAL CONFERENCE

Edited by Dieter Fensel (Vrije Universiteit, The Netherlands), Fausto Giunchiglia (University of Trento, Italy), Deborah McGuinness (Stanford University, USA), and Mary-Anne Williams (The University of Newcastle, Australia)

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 have passed a stringent review process and cover a wide range of topics including: representational formalisms; reasoning techniques; implemented KR&R systems; significant applications for machine learning, decision theory, uncertainty, databases, software engineering, planning, robotics, intelligent agents, and the Semantic Web.

These proceedings are an essential reference volume for researchers and students interested in a detailed view of this key research area.

ISSN 1046-9567
Artificial Intelligence

Morgan Kaufmann Publishers
340 Pine Street, 6th Floor
San Francisco, CA 94104
http://www.mkp.com

ISBN 1-55860-847-8

9 781558 608474 90000